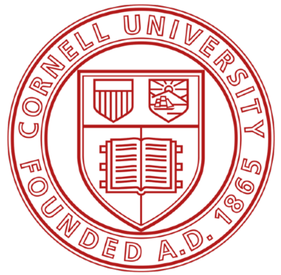
Lecture 15: More diffusion

CS 5788: Introduction to Generative Models



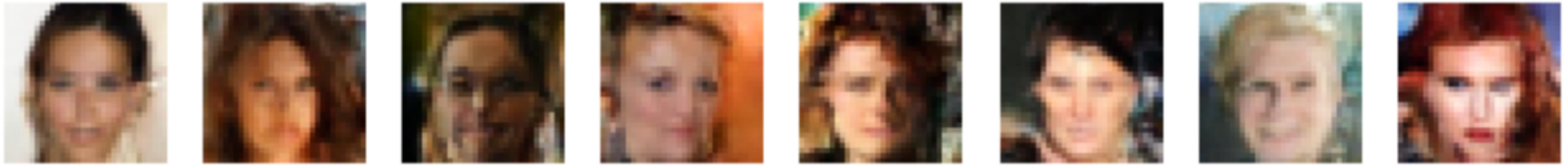
Many figures and most explanation from [Ho et al., “DDPM”, 2020]

Today

- Probabilistic model for diffusion
- Where does the denoising loss come from?
- Speeding up diffusion models

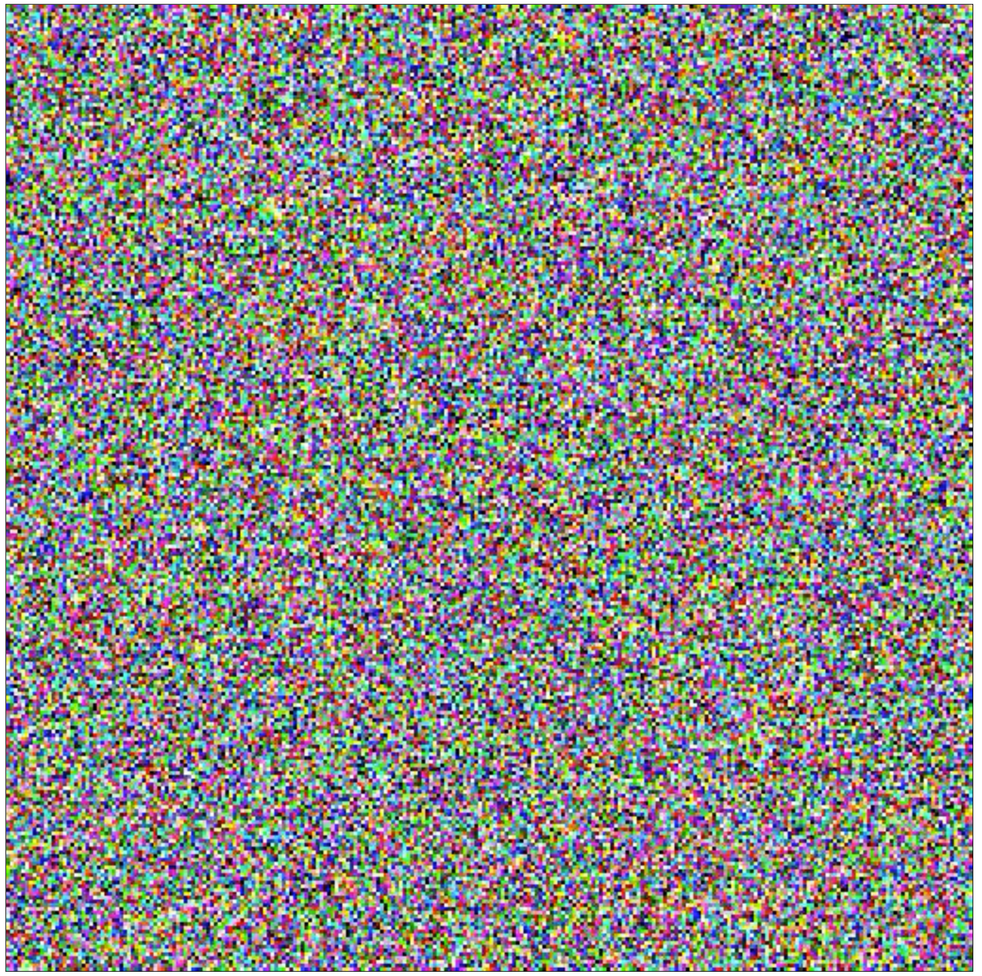
PS3: diffusion models and normalizing flow

- Normalizing flow model (RealNVP)



- Diffusion models

Recall: denoising diffusion probabilistic models (DDPMs)



In a previous class, we gave an intuitive description of diffusion models as denoisers. Let's look more carefully into why they work.

Example source: Aditya Ramesh

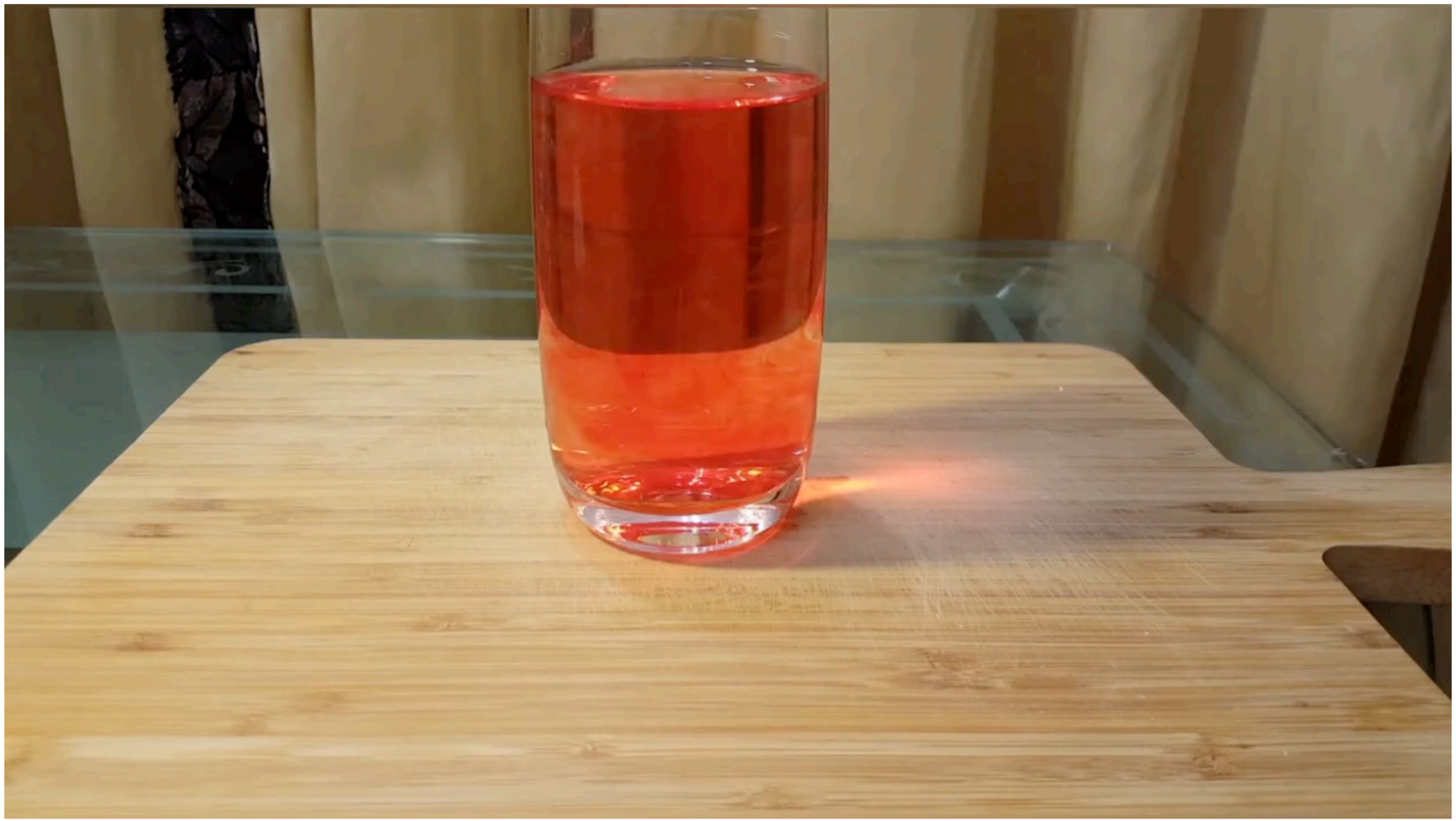
Physical motivation for diffusion

Heat Diffusion

Source: Angjoo Kanazawa

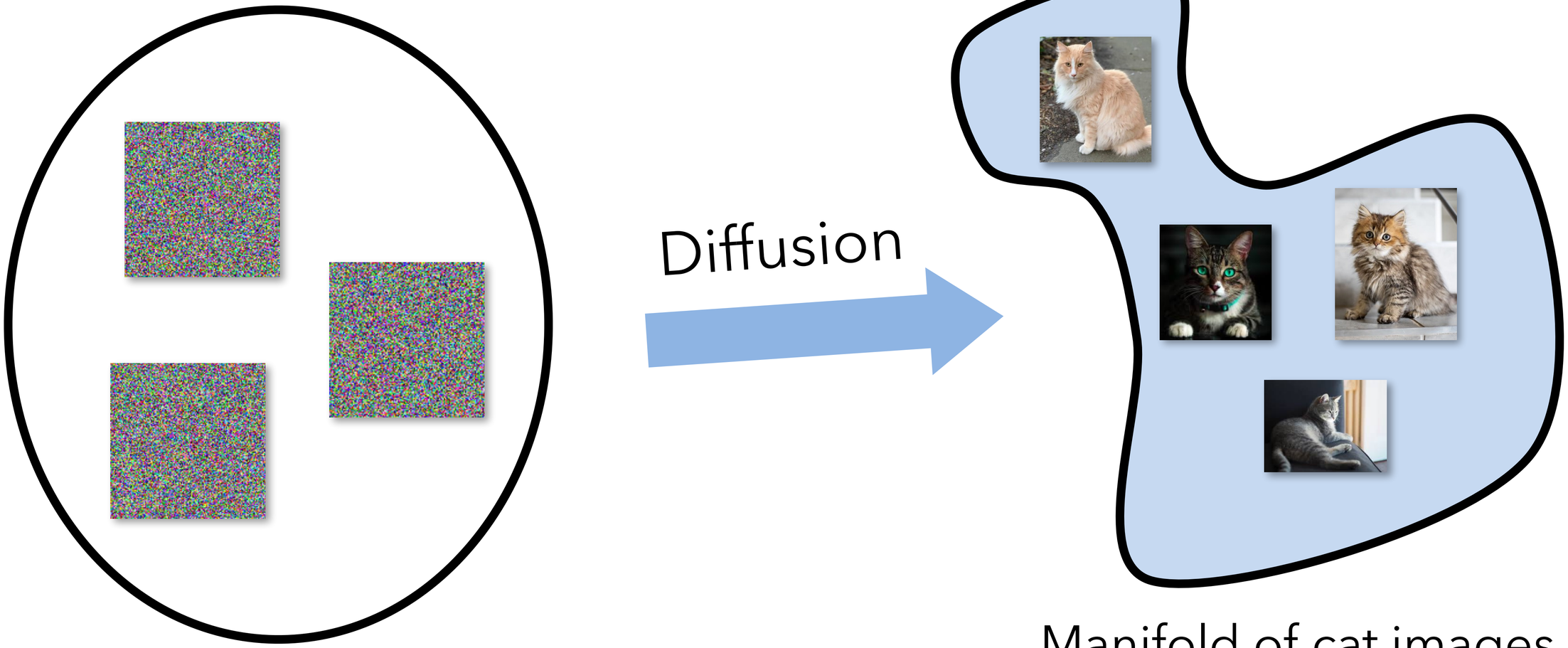
Physical motivation for diffusion

Reversing the process

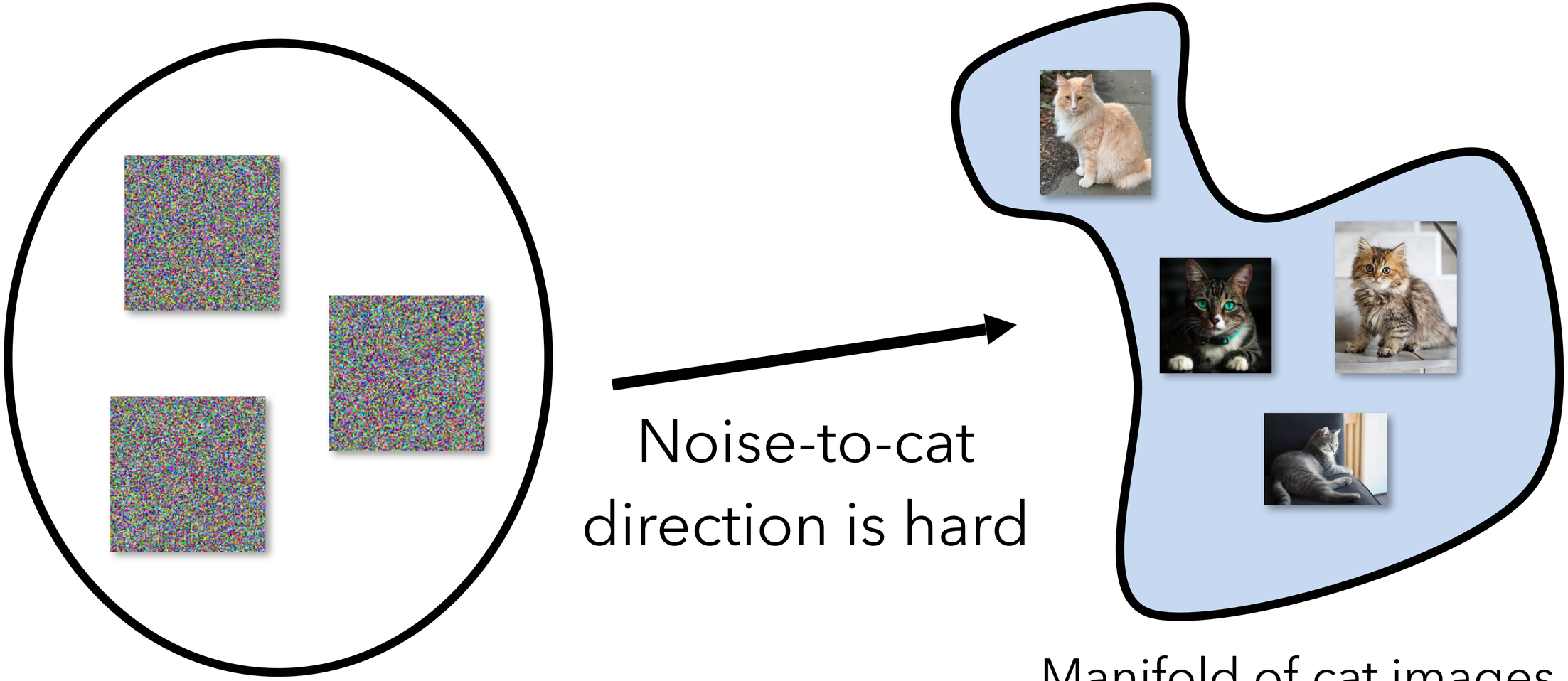


Source: Angjoo Kanazawa

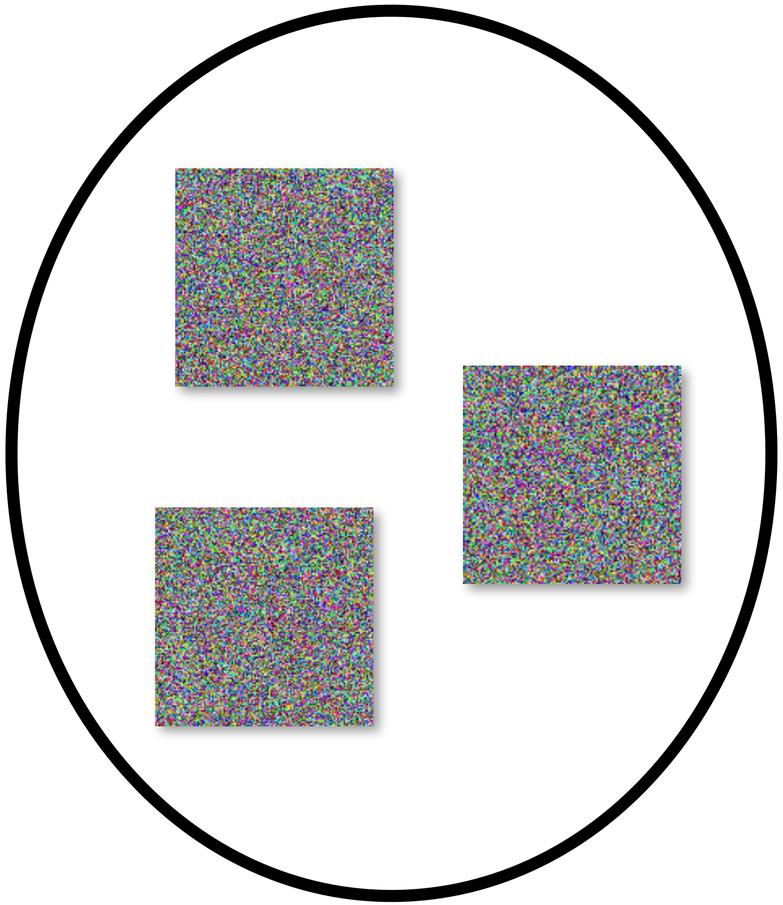
Idea in Song et al. Score-Based Generative Modeling through Stochastic Differential Equations 2021



Slide source: N. Snavely & S. Seitz

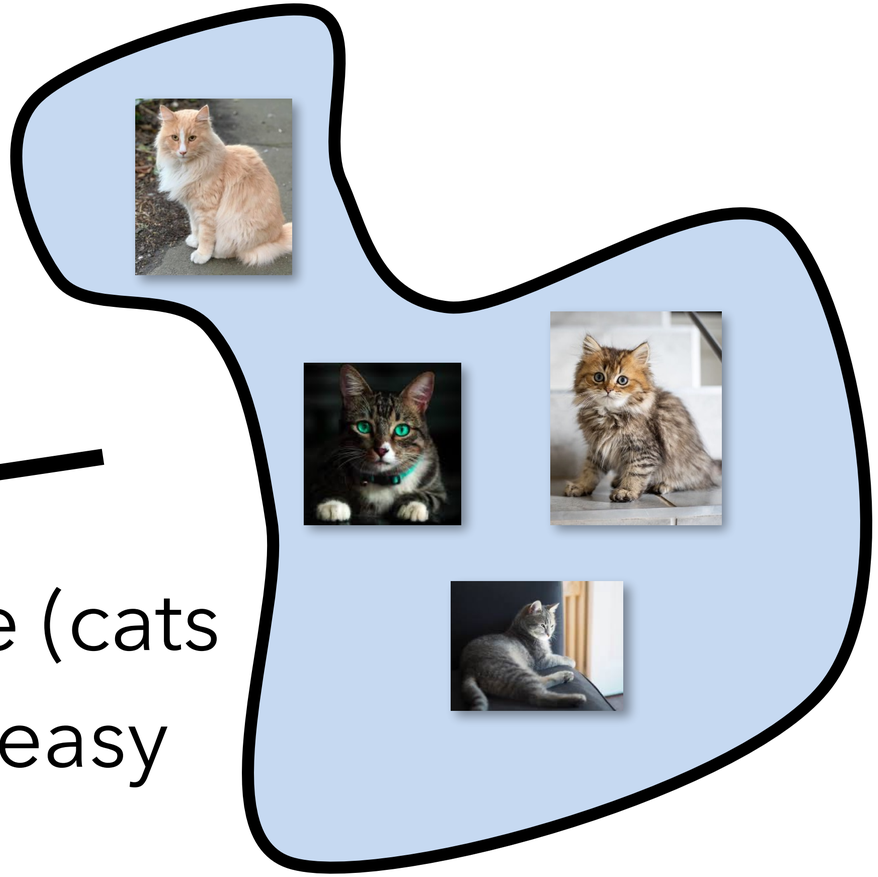


Slide source: N. Snavely & S. Seitz



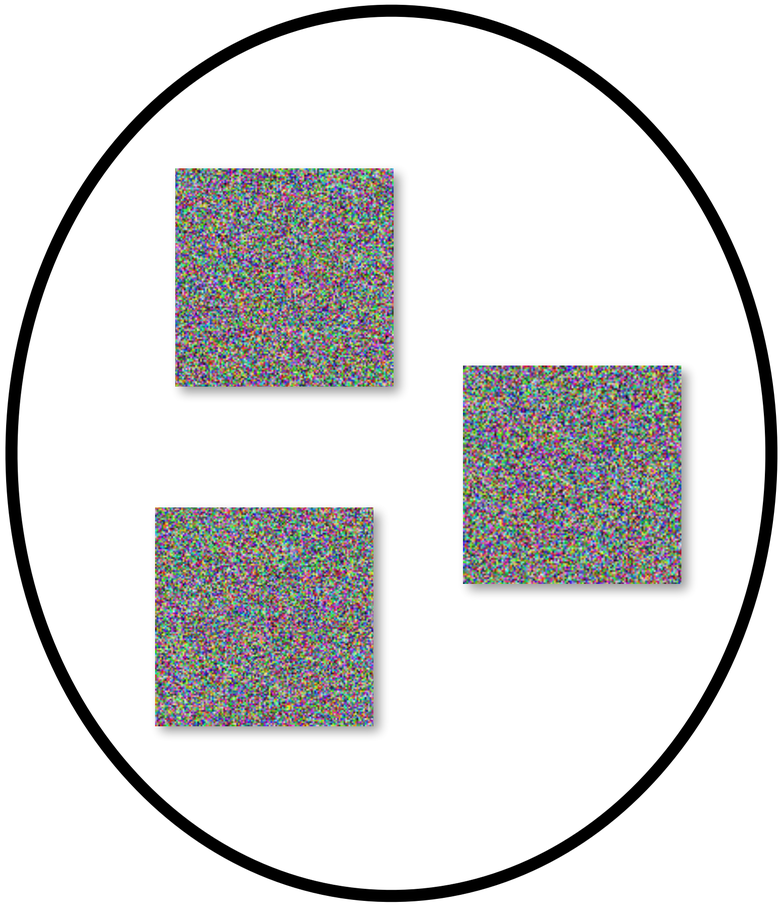
Random images

The opposite (cats to noise) is easy

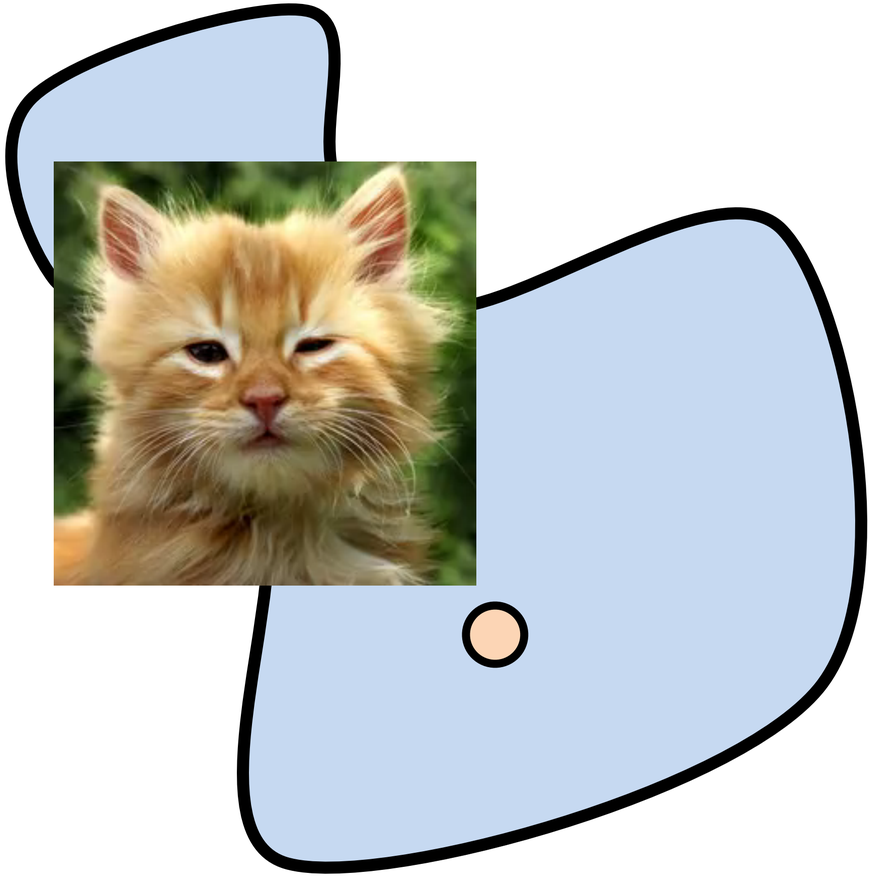


Manifold of cat images

Slide source: N. Snavely & S. Seitz

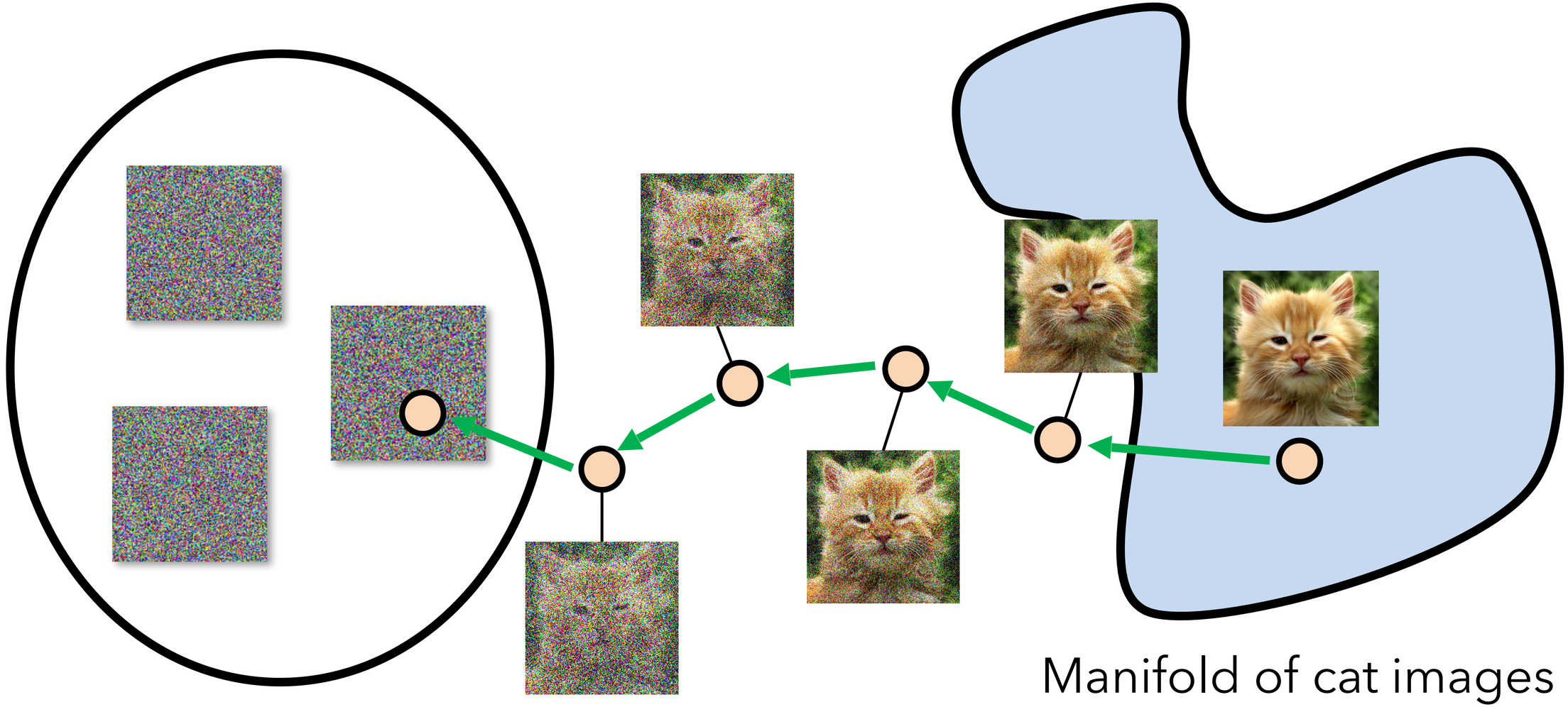


Random images

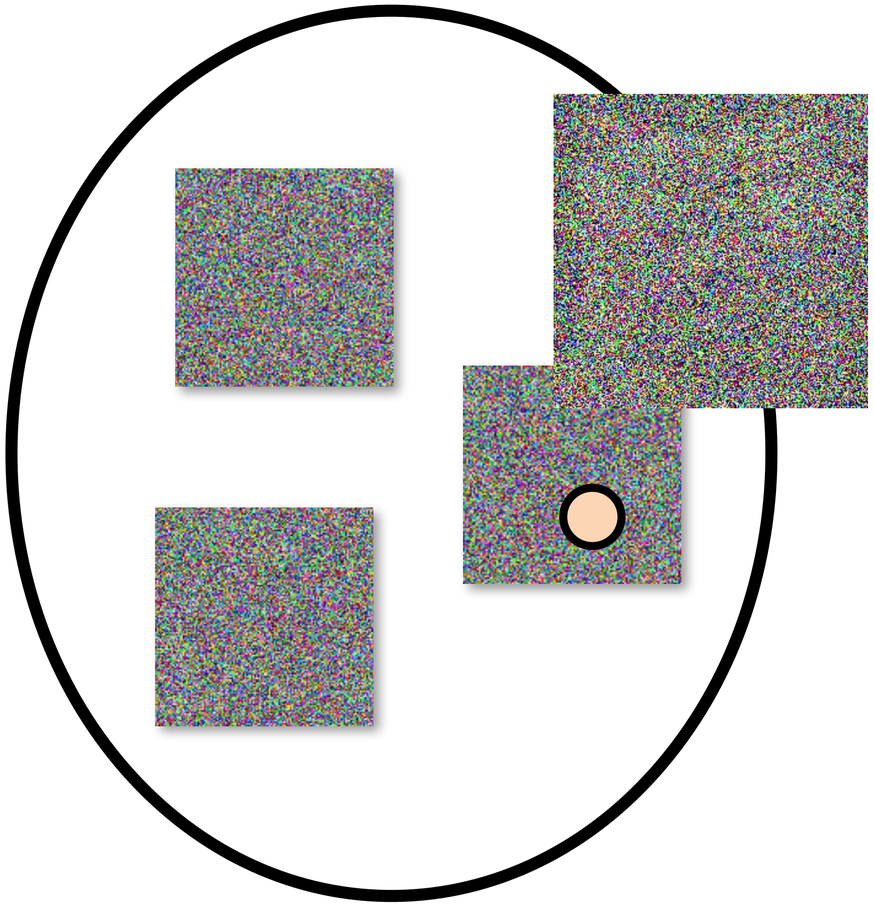


Manifold of cat images

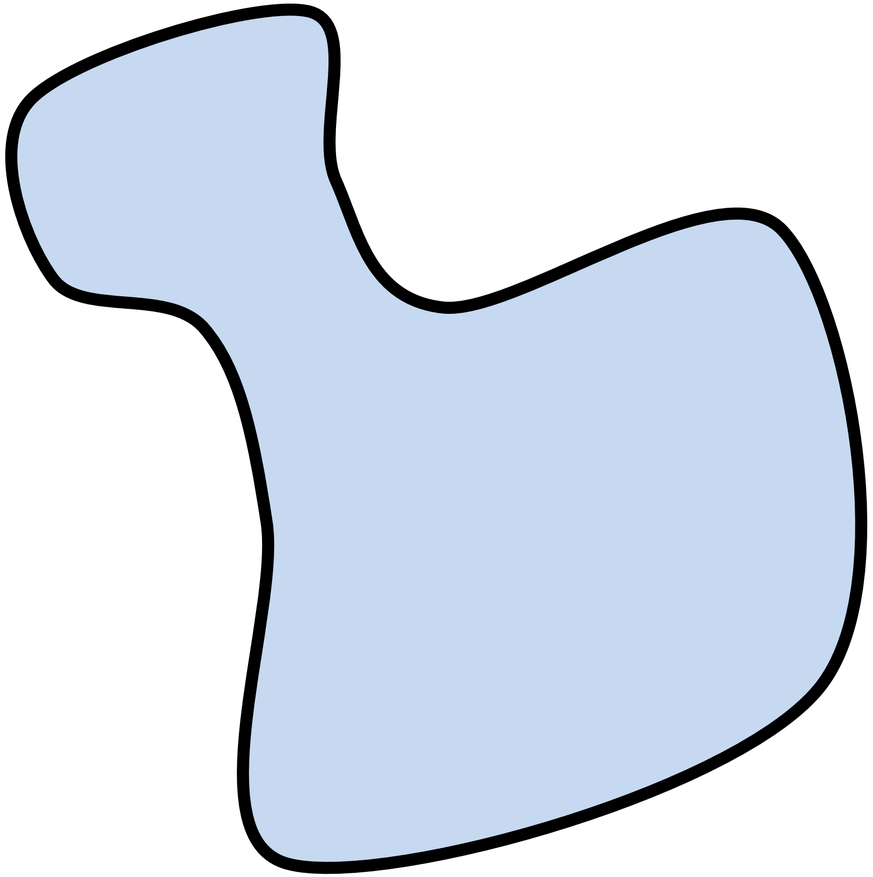
Slide source: N. Snavely & S. Seitz



Slide source: N. Snavely & S. Seitz

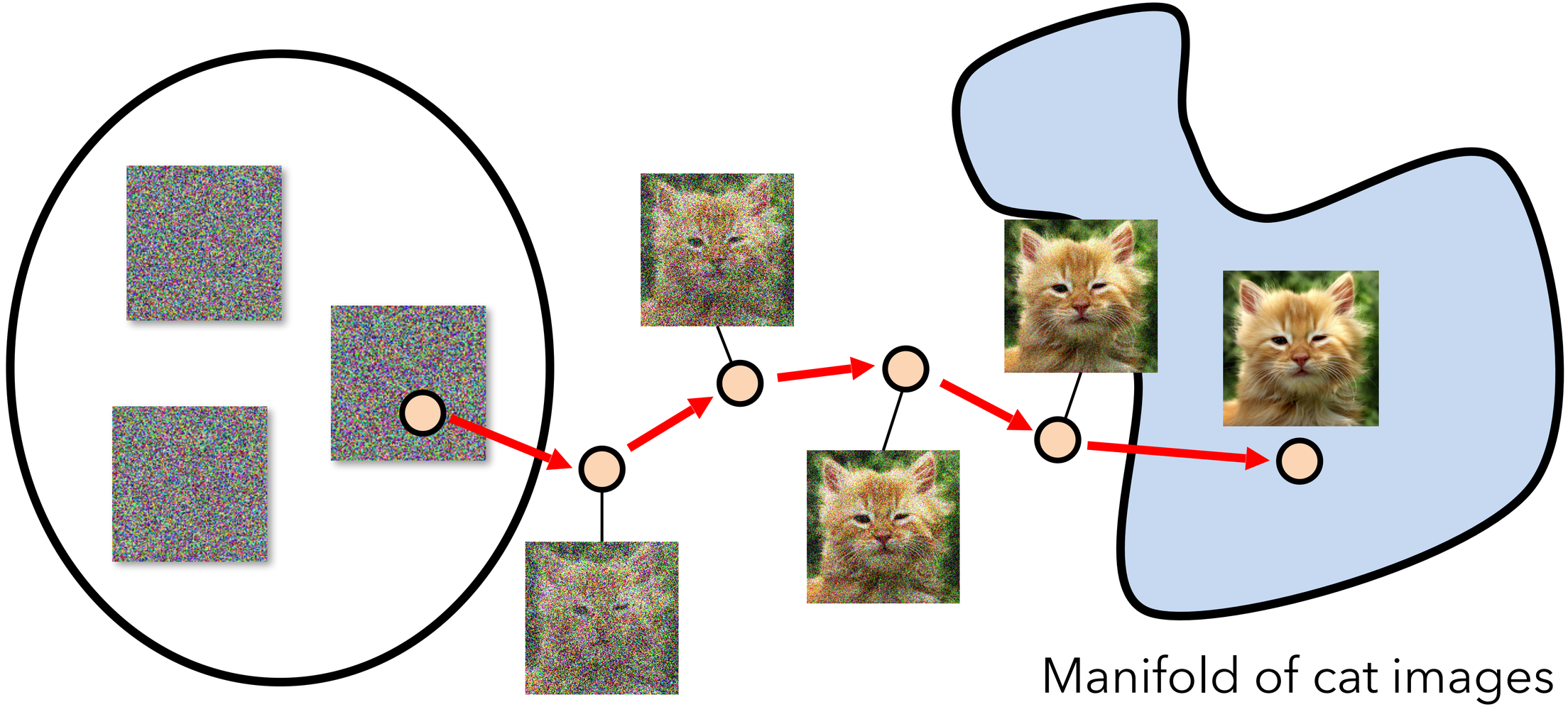


Random images

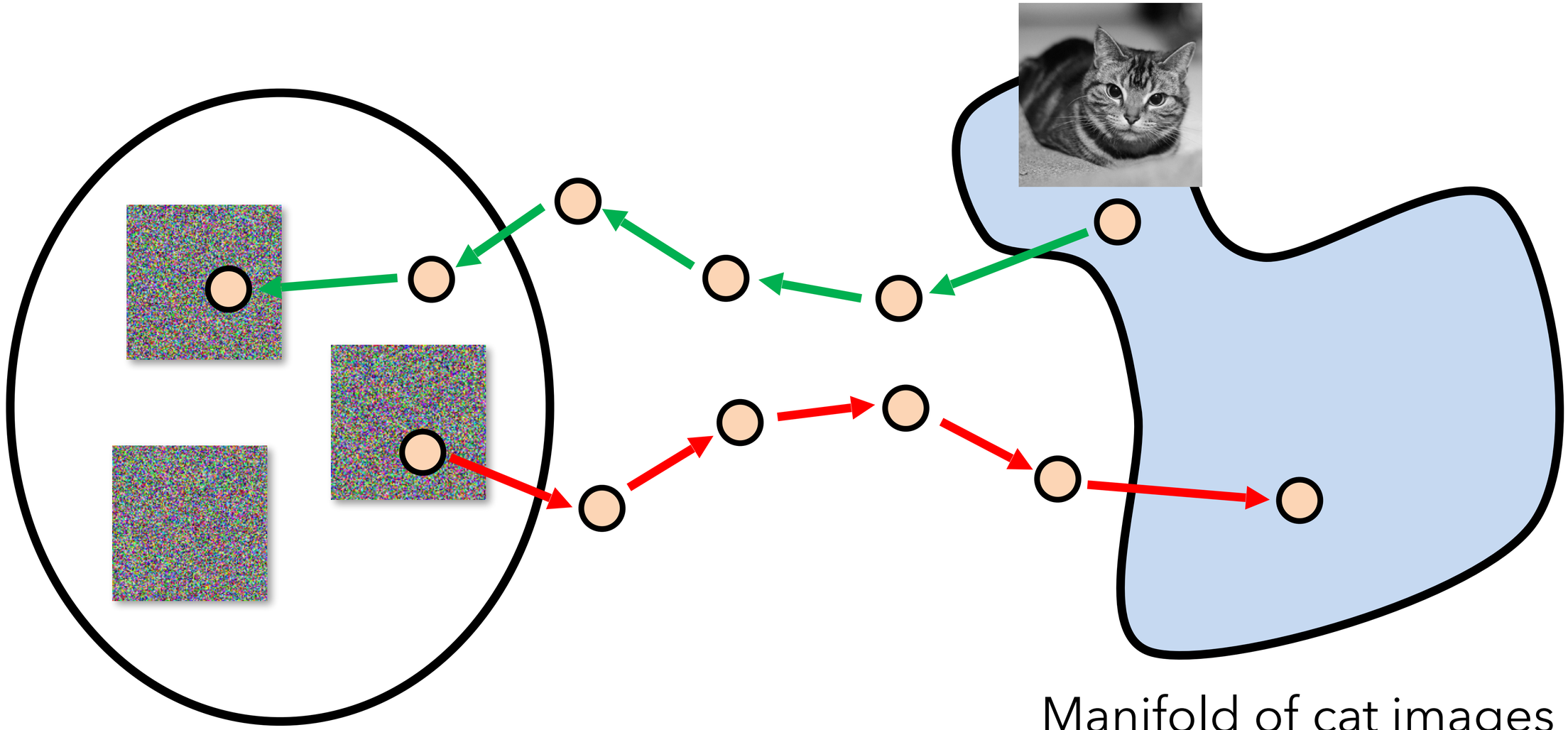


Manifold of cat images

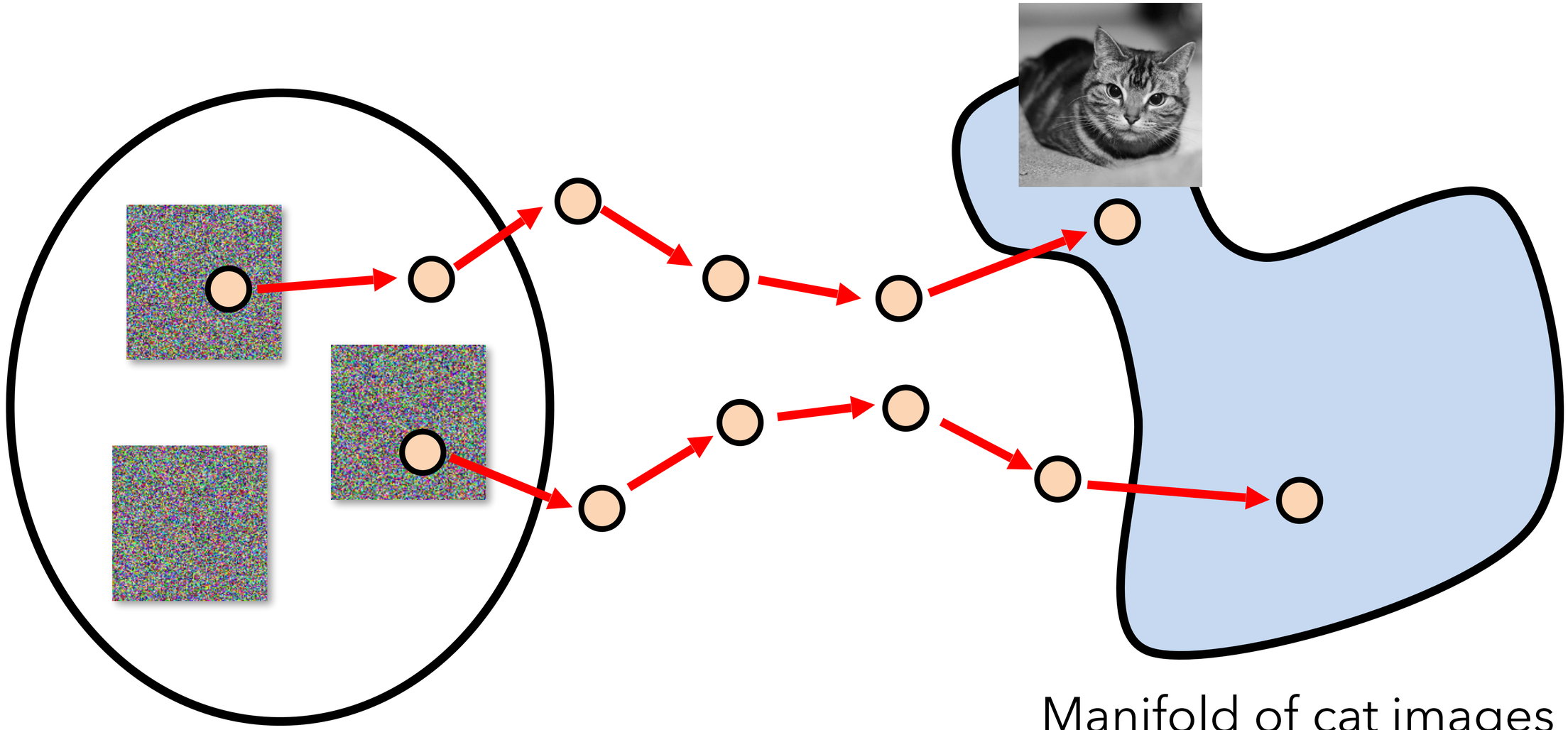
Slide source: N. Snavely & S. Seitz



Slide source: N. Snavely & S. Seitz



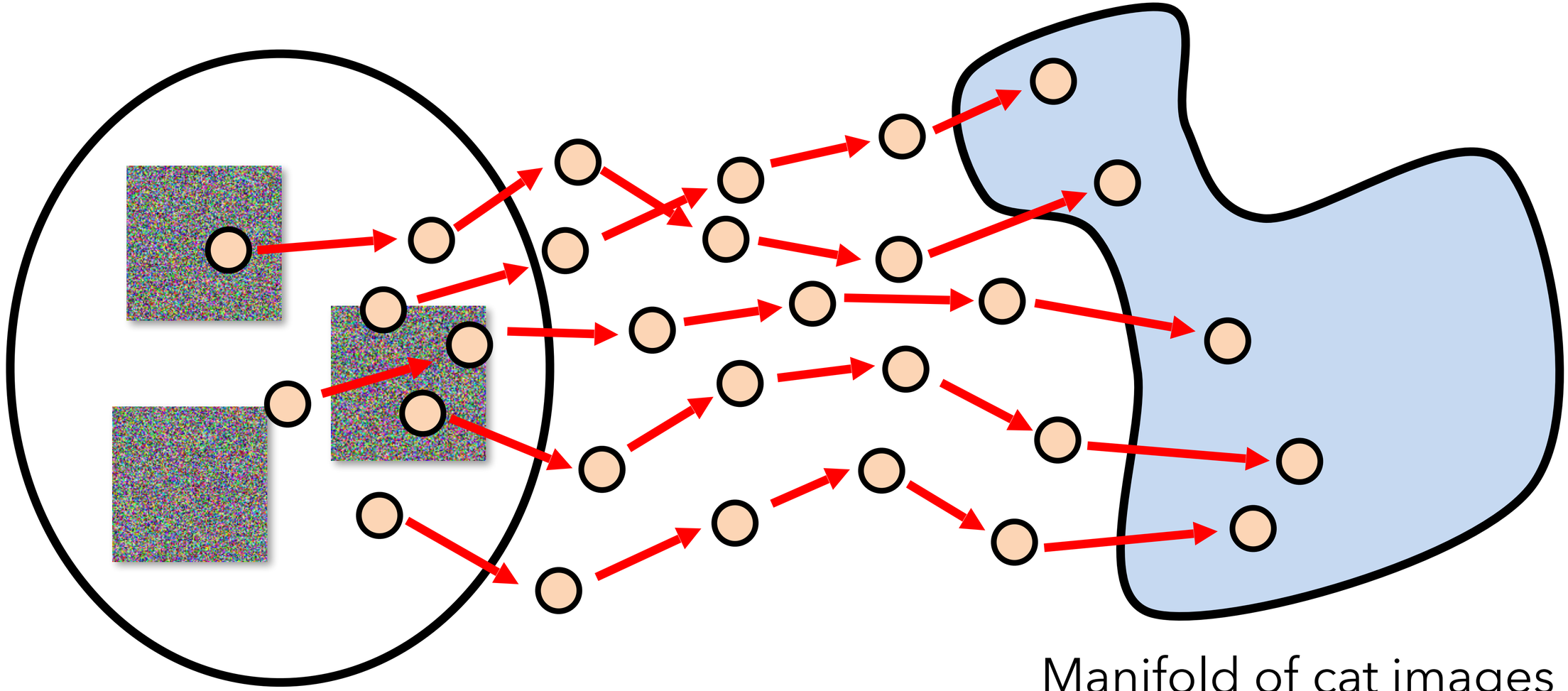
Slide source: N. Snavely & S. Seitz



Random images

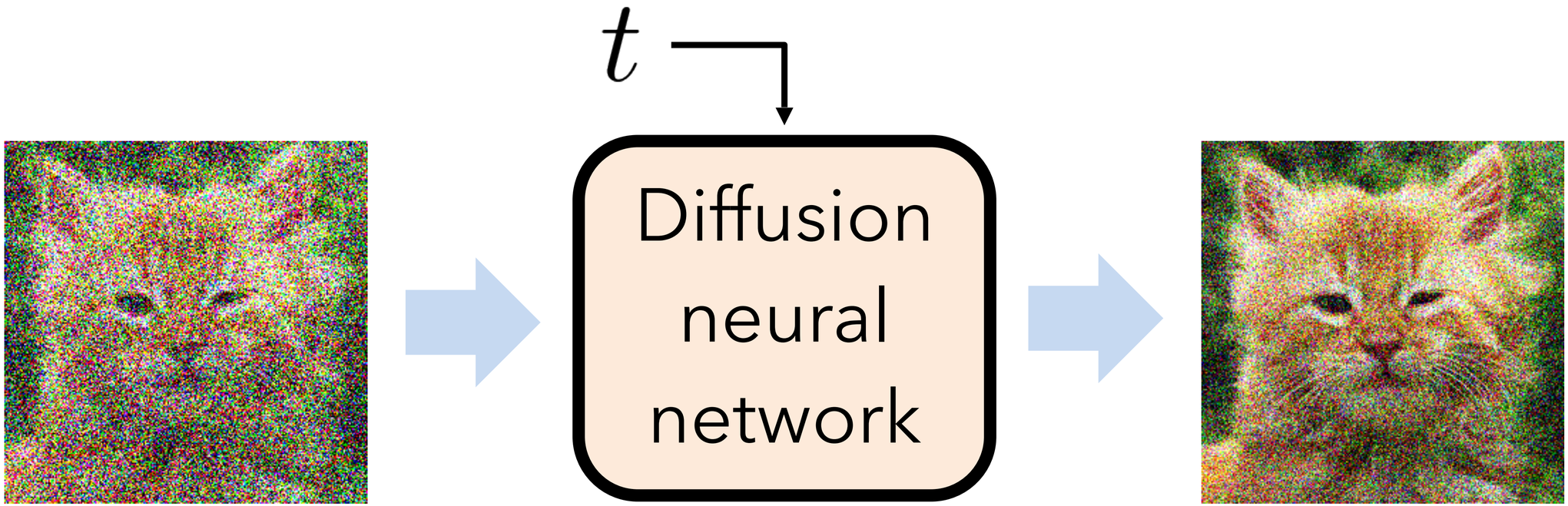
Manifold of cat images

Slide source: N. Snavely & S. Seitz



Slide source: N. Snavely & S. Seitz

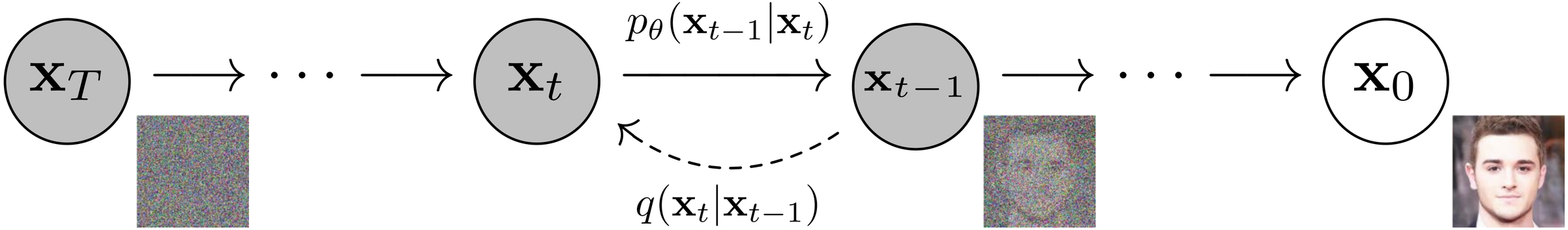
Denoising diffusion neural network



A basic diffusion approach will run this denoising for many timesteps (e.g., T = 1000 steps)

Slide source: N. Snavely & S. Seitz

Recall: training a diffusion model



Algorithm 1 Training

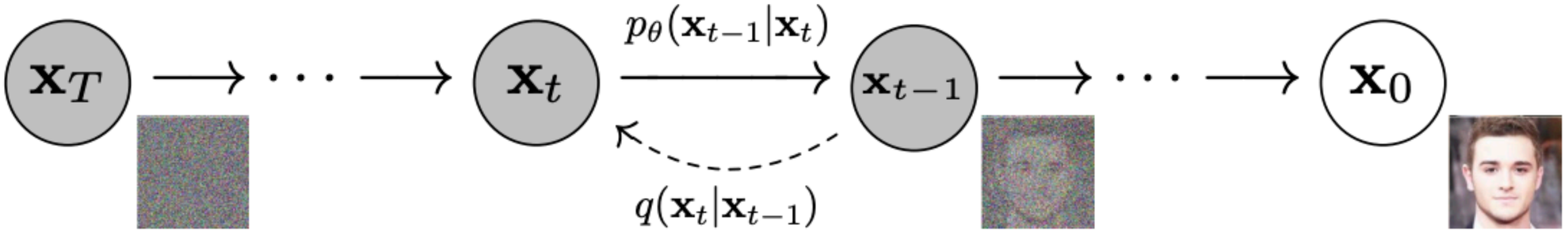
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

6: **until** converged

Figure from [J. Ho et al, "DDPM", 2020]
Adapted from Liyue Shen

Recall: sampling from a diffusion model



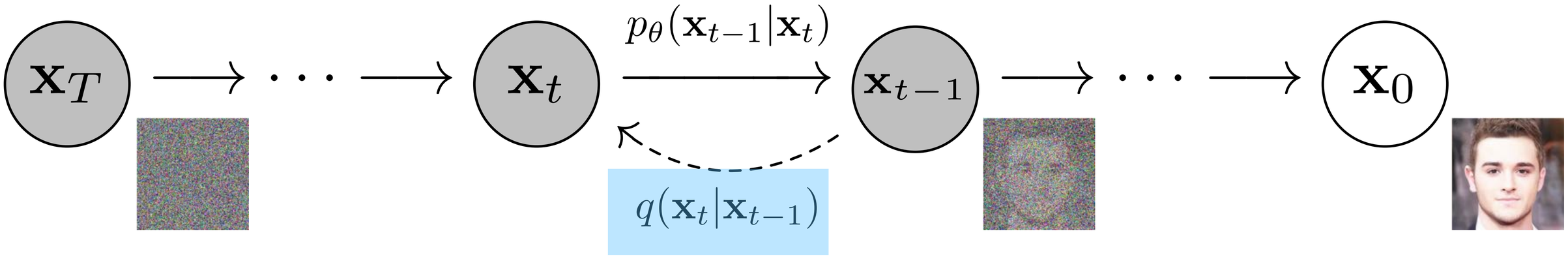
Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Figure from [J. Ho et al, "DDPM", 2020]
Adapted from Liyue Shen

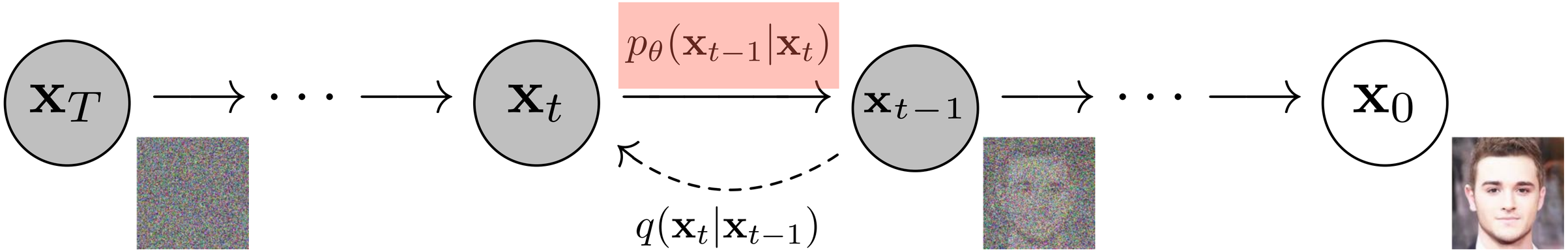
Why does this work?

Diffusion model formulation



- Forward process (adding noise and scaling):
  $q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)I)$
- Coefficients α_t define a schedule (typically these are fixed and hand-chosen).

Diffusion model formulation



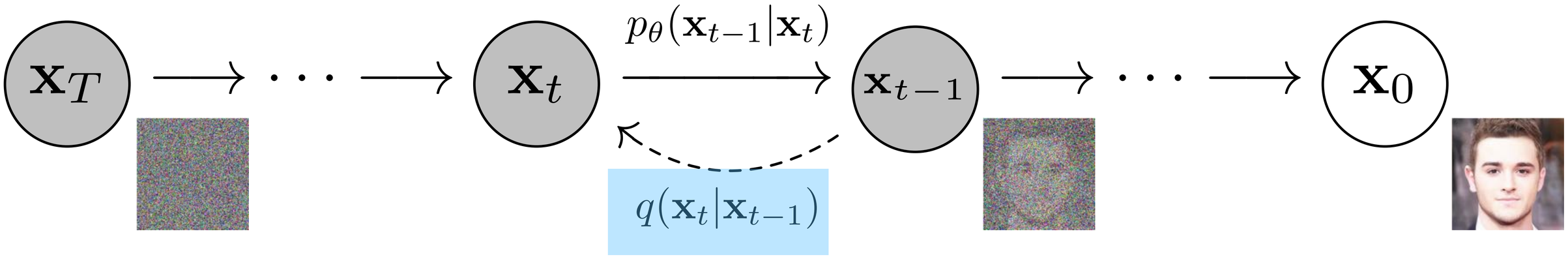
- Backwards process (removing noise and scaling):

  $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t))$
- Learned and usually parametrized by a neural net.
- For simplicity, can set $\boldsymbol{\Sigma}_\theta(\mathbf{x}) = (1 - \alpha_t)I$

Why does this work?

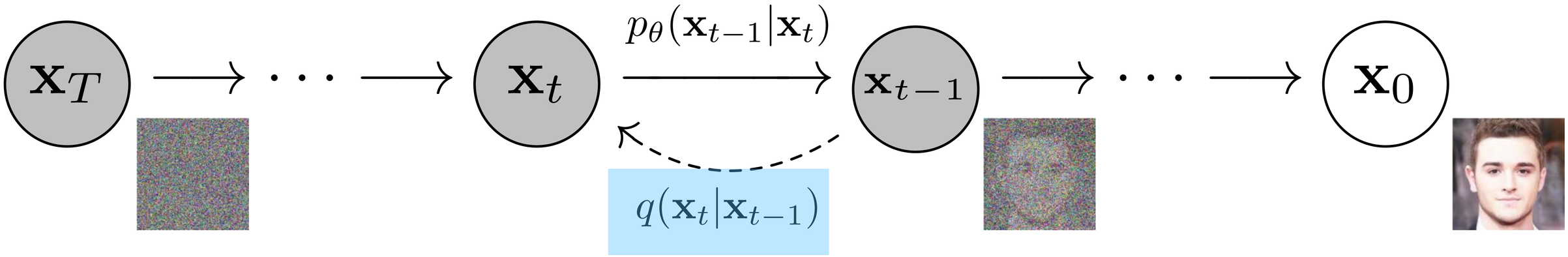
- How do we train this model?
- Why does this probabilistic model lead to the denoising approach that we motivated at the beginning of class?

Preliminaries: Markov chains



- Both p_θ and q have the *Markov property*. The current state depends only on previous state.
- In other words, knowing the earlier history doesn't help!
  - $q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = q(\mathbf{x}_t \mid \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{t-1})$
  - $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_T, \mathbf{x}_{T-1}, \ldots, \mathbf{x}_t)$

Preliminaries: Markov chains



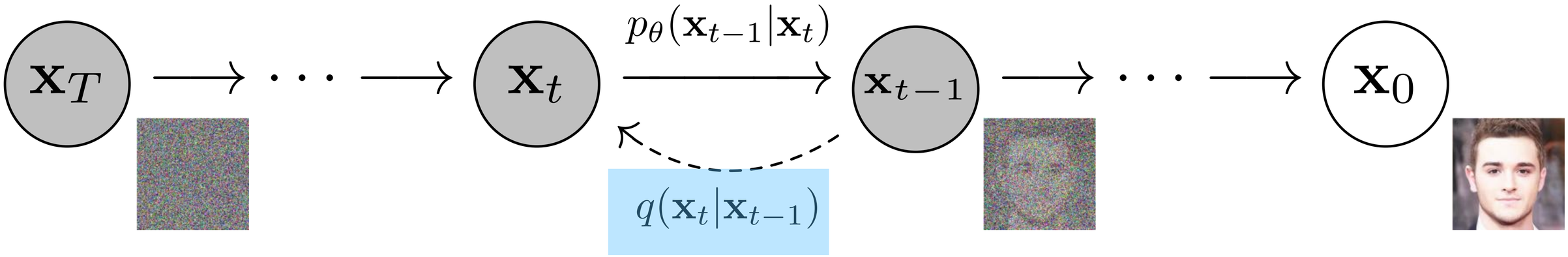
- Consequently, the densities factorize:

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$$

$$\text{and } q(\mathbf{x}_{0:T}) = p(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T) = q(\mathbf{x}_0) \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$$

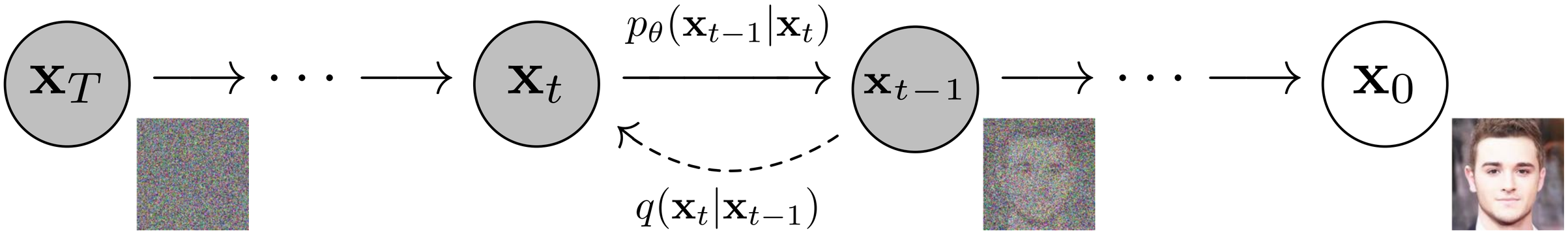
- Why doesn't $p(\mathbf{x}_T)$ depend on θ?

The forward process



- If we want to compute $\mathbf{x}_t$ from $\mathbf{x}_0$, we can do it incrementally
  - $\mathbf{x}_1 \sim q(\mathbf{x}_1 \mid \mathbf{x}_0)$
  - $\mathbf{x}_2 \sim q(\mathbf{x}_2 \mid \mathbf{x}_1)$
  - ...
  - $\mathbf{x}_t \sim q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$

The forward process



- But since q is Gaussian, we can also jump directly from $\mathbf{x}_0$ in a single step:

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon_t \text{ for } \epsilon = \mathcal{N}(0,I)$$

$$= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\epsilon$$

$$\cdots$$

$$= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$$

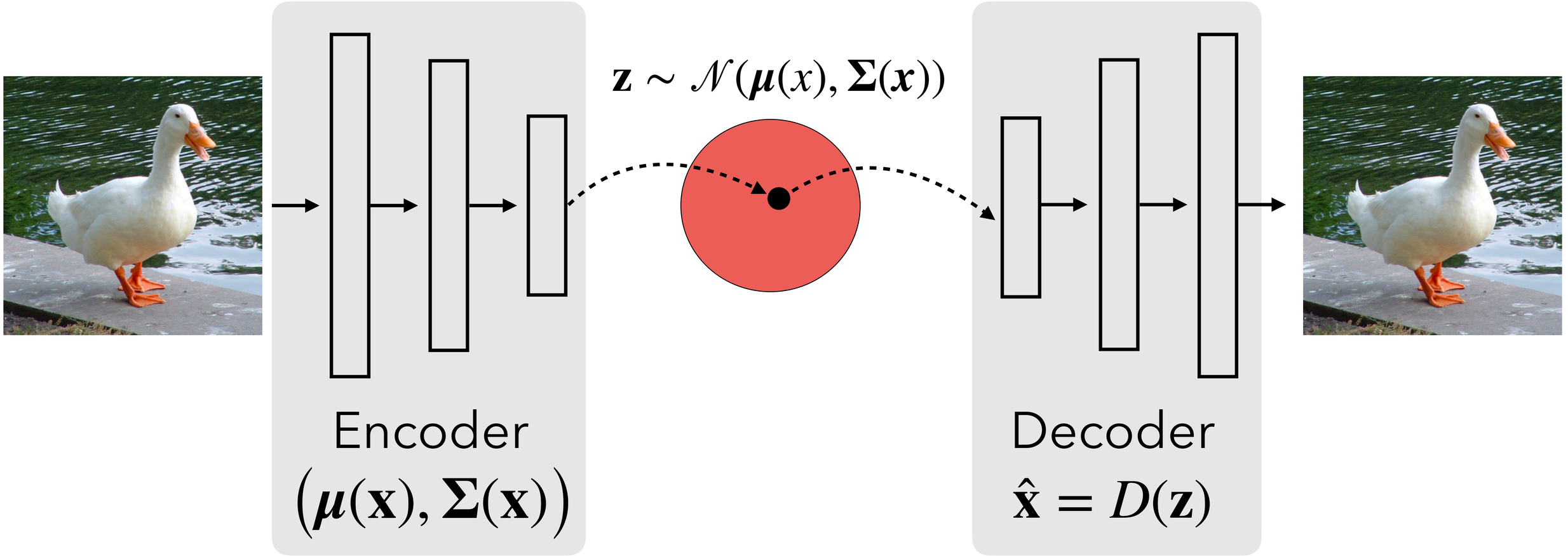
Therefore:

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t;\ \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)I)$$

Adapted from V. Kuleshov

How do we train the model?

Analogy to variational autoencoder



- Encoder net defines a distribution over $\mathbf{z}$: $q(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}\left(\boldsymbol{\mu}(x), \boldsymbol{\Sigma}(\boldsymbol{x})\right)$.
- Decoder net defines a distribution over $\mathbf{x}$: $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\hat{\mathbf{X}}, I)$.

Analogy to VAEs

- Latents are $\mathbf{z} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$
- The "encoder" $q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)$ is hand-defined and not learned.

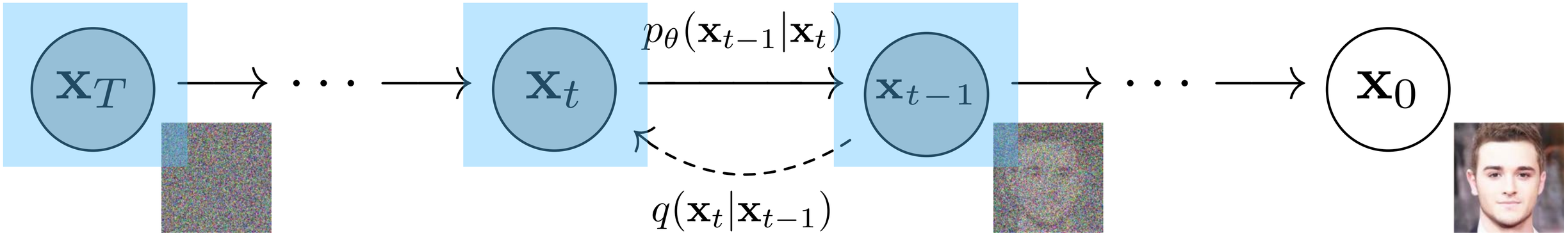


Figure from [J. Ho et al, "DDPM", 2020]

Analogy to VAEs

- Observed data: $\mathbf{x}_0$
- The decoder is p_θ, hierarchically decoding each $\mathbf{x}_t$.

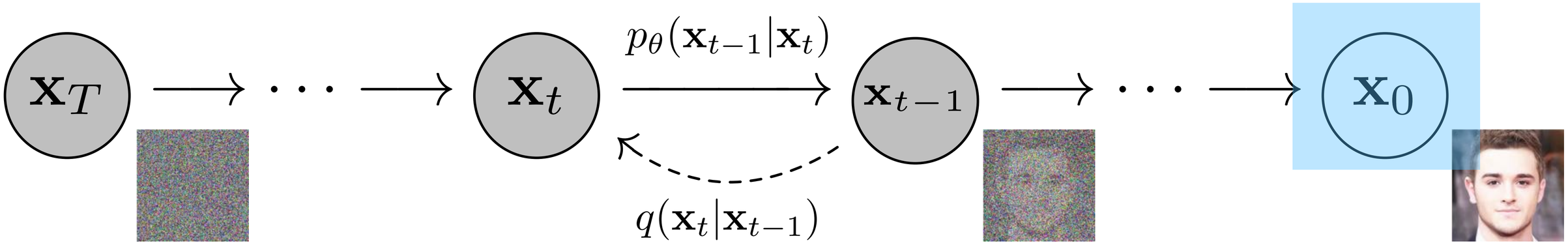


Figure from [J. Ho et al, "DDPM", 2020]

Recall: evidence lower bound

$$\log p(\mathbf{x}) = \log \int p(\mathbf{x}, \mathbf{z})\, d\mathbf{z}$$

in VAE, the encoder, e.g., noise given image

$$= \log \int q(\mathbf{z} \mid \mathbf{x}) \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z} \mid \mathbf{x})}\, d\mathbf{z}$$

Evidence lower bound

$$\log p(\mathbf{x}) = \log \int p(\mathbf{x}, \mathbf{z})\, d\mathbf{z}$$

$$= \log \int q(\mathbf{z} \mid \mathbf{x}) \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z} \mid \mathbf{x})}\, d\mathbf{z}$$

$$\geq \int q(\mathbf{z} \mid \mathbf{x}) \log \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z} \mid \mathbf{x})}\, d\mathbf{z}$$

by Jensen's inequality

$$= \mathbb{E}_q \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q(\mathbf{z} \mid \mathbf{x})} \right]$$

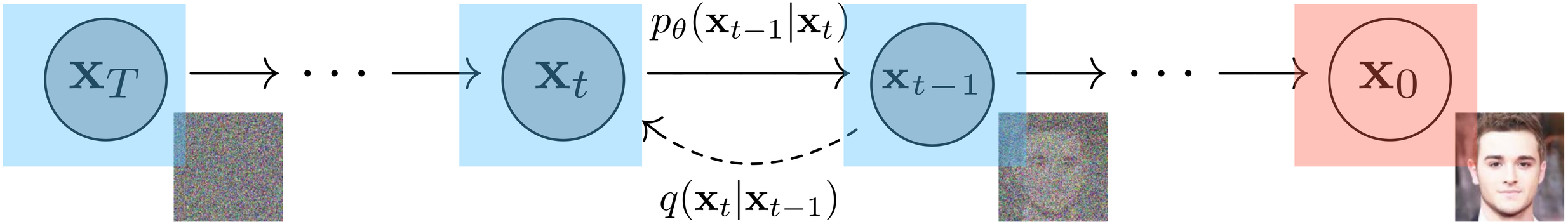
What is the ELBO for diffusion models?

$$\log(p_\theta(\mathbf{x}_0)) \geq \mathbb{E}_q \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q(\mathbf{z} \mid \mathbf{x})} \right]$$

$$= \mathbb{E}_q \left[\log \frac{p_\theta(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T)}{q(\mathbf{x}_1, \ldots, \mathbf{x}_T \mid \mathbf{x}_0)} \right]$$

alternative notation

$$= \mathbb{E}_q \left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} \right]$$



ELBO for diffusion models

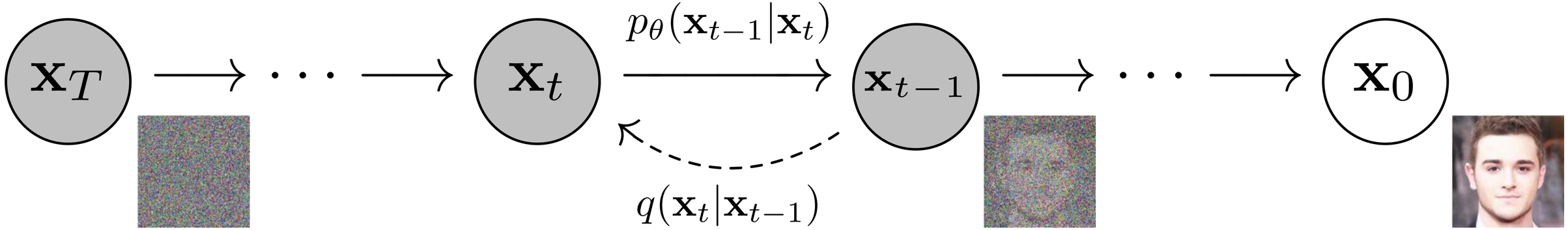
$$\log(p_\theta(\mathbf{x}_0)) \geq \mathbb{E}_q \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q(\mathbf{z} \mid \mathbf{x})} \right]$$

alternative notation

$$= \mathbb{E}_q \left[\log \frac{p_\theta(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T)}{q(\mathbf{x}_1, \ldots, \mathbf{x}_T \mid \mathbf{x}_0)} \right] = \mathbb{E}_q \left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} \right]$$

$$= \mathbb{E}_q \left[\log p(\mathbf{x}_T) + \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_t \mid \mathbf{x}_{t-1})} \right]$$

Markov property



ELBO for diffusion models

$$\log(p_\theta(\mathbf{x}_0)) \;\geq\; \mathbb{E}_q \left[\log p(\mathbf{x}_T) + \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_t \mid \mathbf{x}_{t-1})} \right]$$

In PS3, you'll rewrite this as follows, through simple algebraic manipulation and using the Markov property:

$$-\log(p_\theta(\mathbf{x}_0)) \leq$$

$$\mathbb{E}_q \left[D_{\mathrm{KL}}(q(\mathbf{x}_T \mid \mathbf{x}_0) \parallel p(\mathbf{x}_T)) + \sum_{t \geq 1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)) - \log(p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)) \right]$$

ELBO for diffusion models

$$-\log(p_\theta(\mathbf{x}_0)) \leq \mathbb{E}_q \Big[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T \mid \mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t \geq 1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)) - \underbrace{\log(p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1))}_{L_0} \Big]$$

doesn't depend on θ!

probability of true example given model's "best guess" $\mathbf{x}_1$

Adapted from V. Kuleshov

ELBO for diffusion models

$$-\log(p_\theta(\mathbf{x}_0)) \leq$$

$$\mathbb{E}_q \left[D_{\mathrm{KL}}(q(\mathbf{x}_T \mid \mathbf{x}_0) \parallel p(\mathbf{x}_T)) + \sum_{t \geq 1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)) - \log(p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)) \right]$$

L_{t-1}

key term: how does the learned backward process match the true backward process?

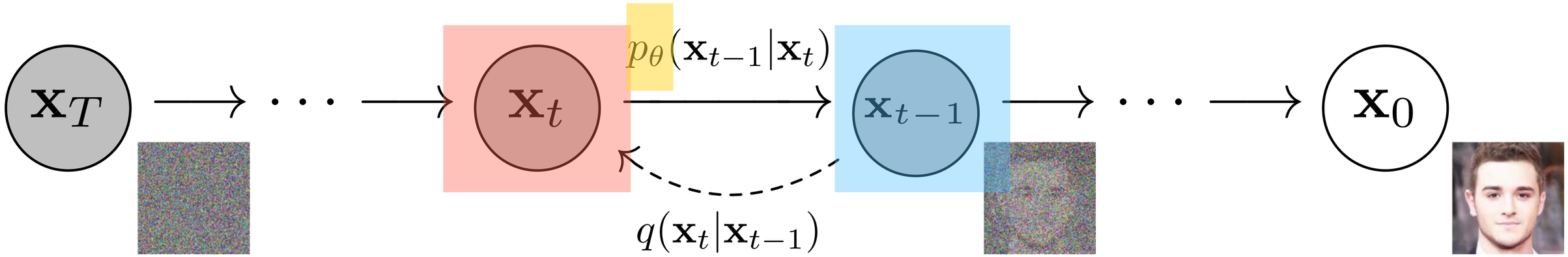
Adapted from V. Kuleshov

ELBO for diffusion models

Let's look more closely at this term:

our model

$$\sum_{t \geq 1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t))$$

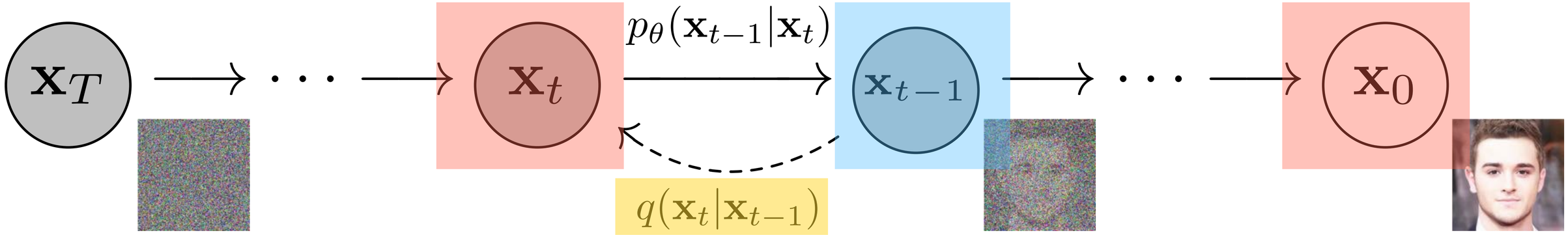


ELBO for diffusion models

Let's look more closely at this term:

$$\sum_{t \geq 1} D_{\mathrm{KL}}\big(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)\big)$$

- If we knew the clean image and the current noisy image, what's the denoised image?
- There's no learning. We can analytically derive this!



ELBO for diffusion models

Let's look more closely at this term:

$$\sum_{t \geq 1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t))$$

We know that $q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ is Gaussian. But $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$ is Gaussian, too!

We can show:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I})$$

where $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t$ and $\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$

Source: [J. Ho et al, "DDPM", 2020]

ELBO for diffusion models

So, this means that:

$$\sum_{t \geq 1} D_{\mathrm{KL}}\big(\underbrace{q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)}_{\text{Gaussian}} \;\|\; \underbrace{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}_{\text{also Gaussian (we defined it that way)}}\big)$$

per-timestep constant

neural net

$$\mathcal{N}(\boldsymbol{\mu}_\theta(\mathbf{x}_t), \sigma_t^2 I)$$

We know the KL divergence between two Gaussians:

$$D_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{\mu}_1, \sigma_t^2 I) \;\|\; \mathcal{N}(\boldsymbol{\mu}_2, \sigma_t^2 I)) = \frac{1}{2\sigma_t^2} \;\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|^2$$

Using this fact, we can obtain a squared loss:

$$D_{\mathrm{KL}}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \;\|\; p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)) = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2 \right] + C$$

A squared loss

So, this means that:

$$\sum_{t \geq 1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t))$$

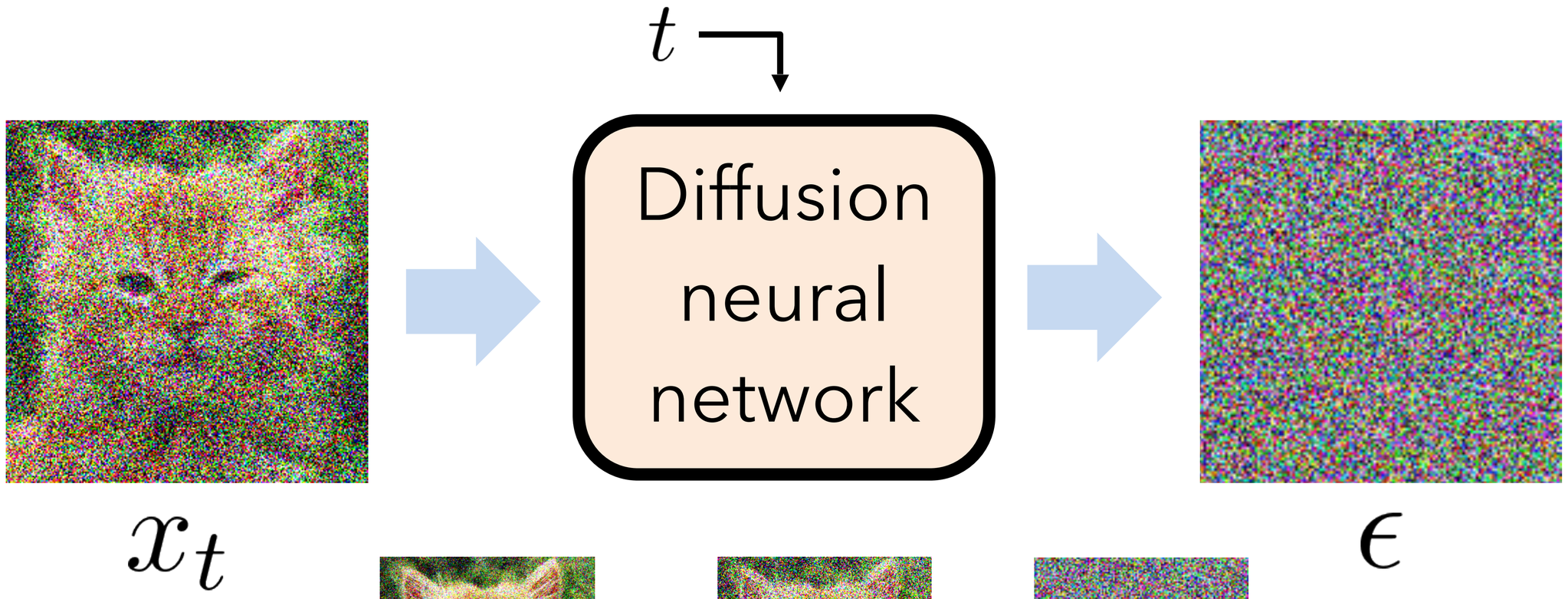
recall: weighted sum of $\mathbf{x}_0$ and $\mathbf{x}_t$

$$= \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2 \right] + C$$

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \quad \text{and} \quad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

Source: [J. Ho et al, “DDPM”, 2020]

Recall: in practice, we can predict the noise itself



Simplifying the loss

If we parameterize the model as a noise predictor: $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$, then through algebraic manipulation, we can show:

$$D_{\mathrm{KL}}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)) = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2 \right] + C$$

scale determined by t

pure noise

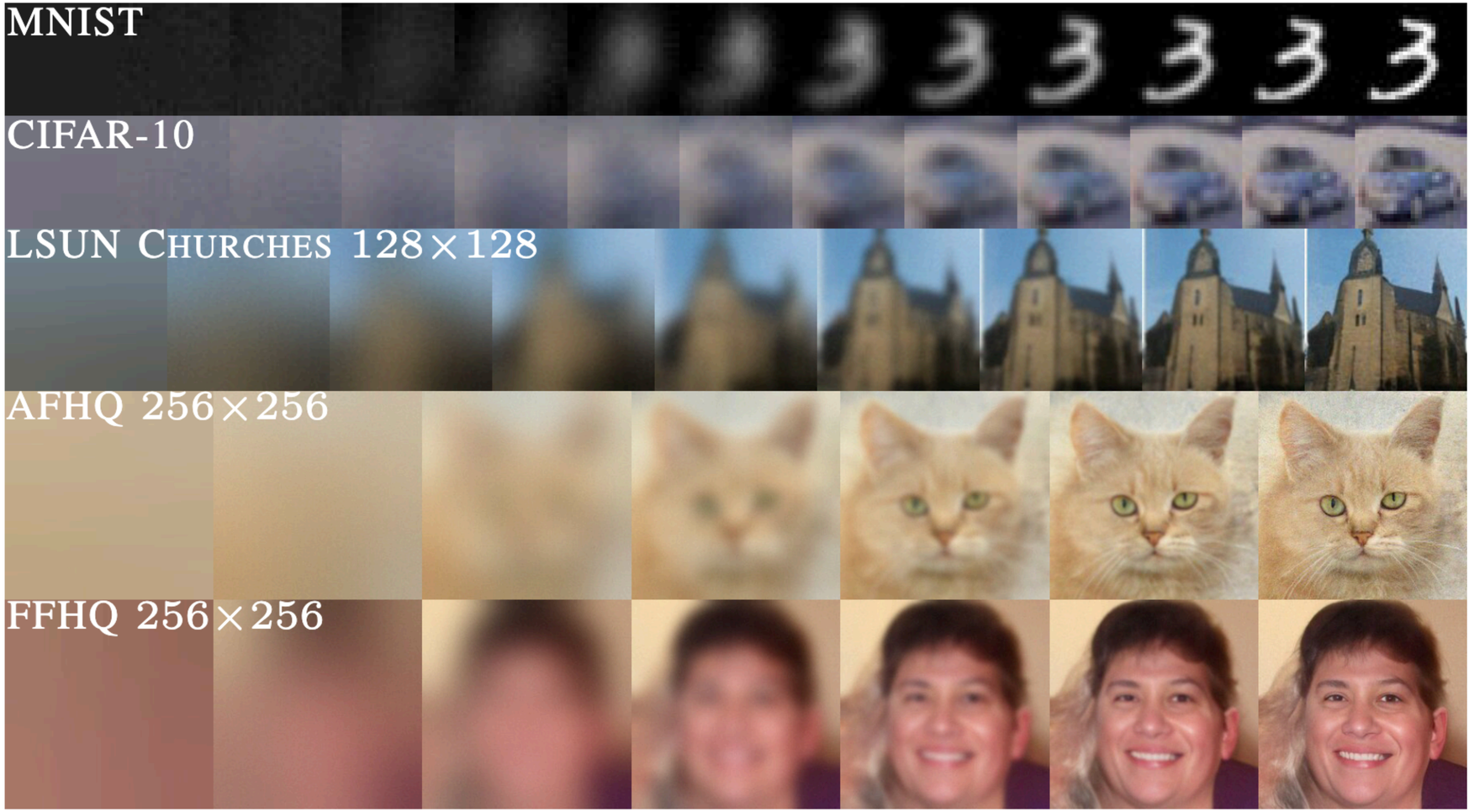
$\mathbf{x}_t$

$$= \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}} \left[\lambda_t \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t \right) \right\|^2 \right]$$

For image generation, people often set $\lambda_t = 1$, which empirically seems to produce better results. This loss is often called L_{simple}.

Revisiting assumptions

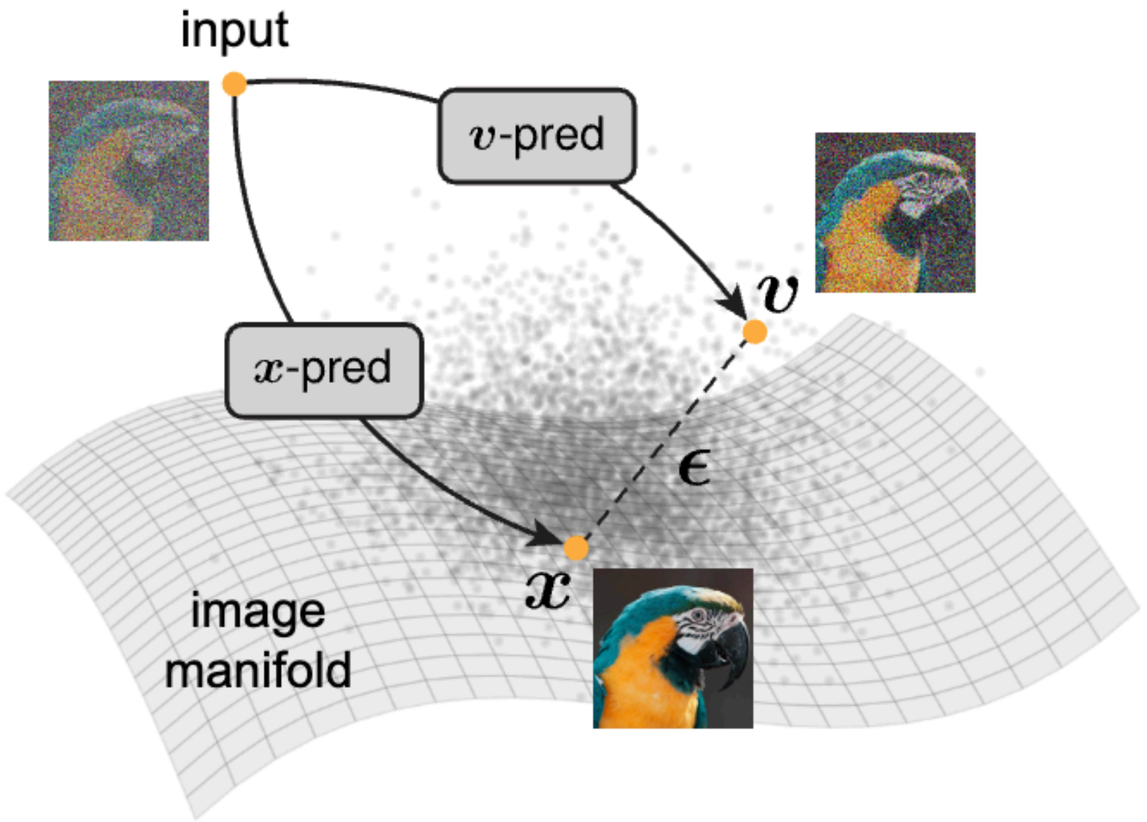
Does the noise distribution need to be Gaussian?



You can derive models that use other distributions (e.g., blurring).

[Rissanen et al., 2023]

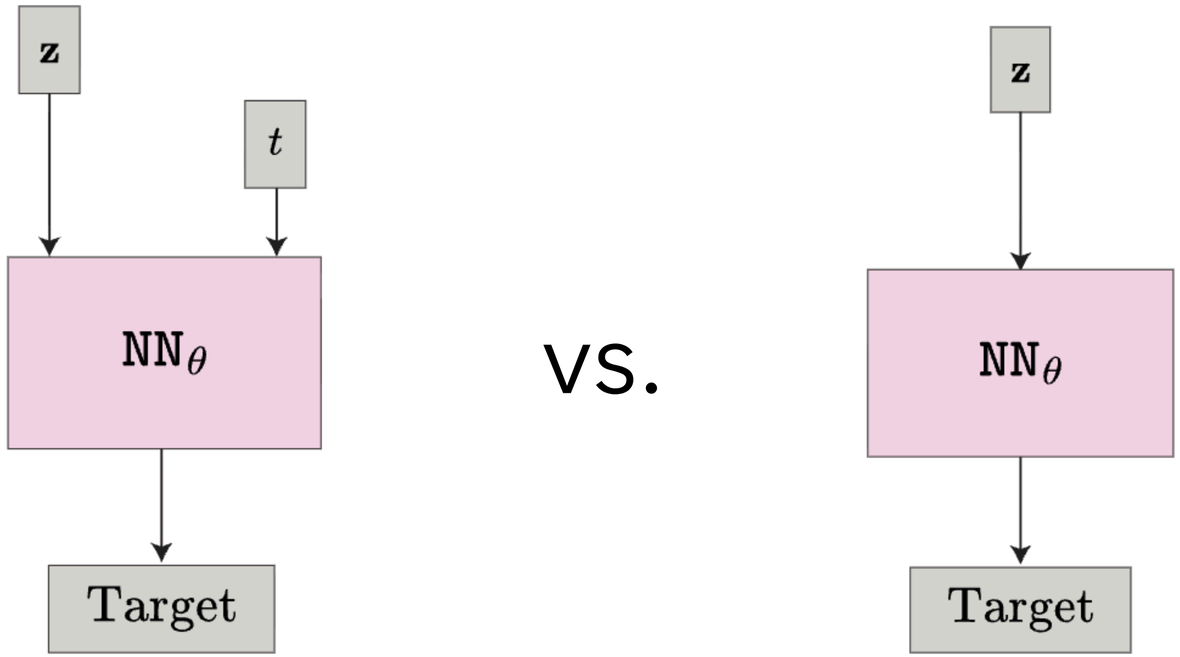
Do you need to predict the noise?



Recent work suggests that predicting the clean example has advantages, since the output space lies in a clean image manifold (whereas with noise, there is no structure).

Source: [Li & He, Back to Basics: Let Denoising Generative Models Denoise, 2026]

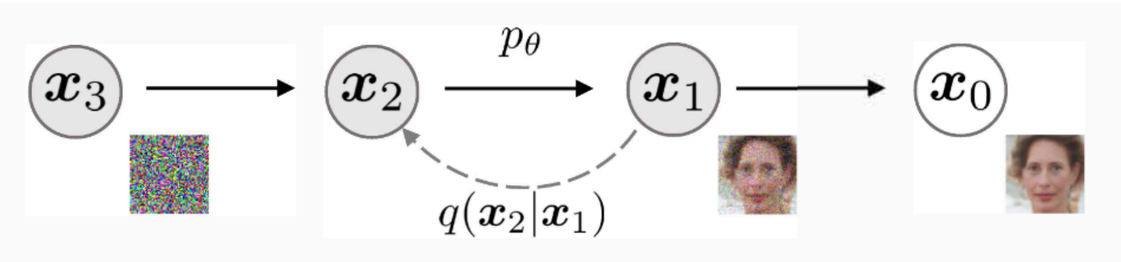
Do we need to condition on t?



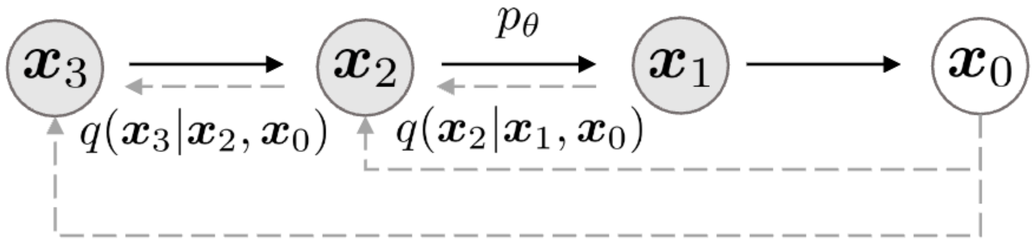
You can often do just as well without explicitly conditioning on t. This makes sense, because you can often denoise images without knowing exactly how much noise was added in practice.

Source: [Sun et al., “Is Noise Conditioning Necessary for Denoising Generative Models?”, 2025]

Do we need to add noise in each reverse step?



Markovian noise



non-Markovian noise

Use forward process (DDIM) defined as:

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\tilde{\sigma}_t^2}\,\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}},\ \tilde{\sigma}_t^2\mathbf{I}\right)$$

Setting $\tilde{\sigma}_t = 0$ results in a deterministic reverse process.

Can be used out-of-the-box with an existing DDPM model.

[Song et al., “DDIM”, 2022]

Speeding up diffusion models

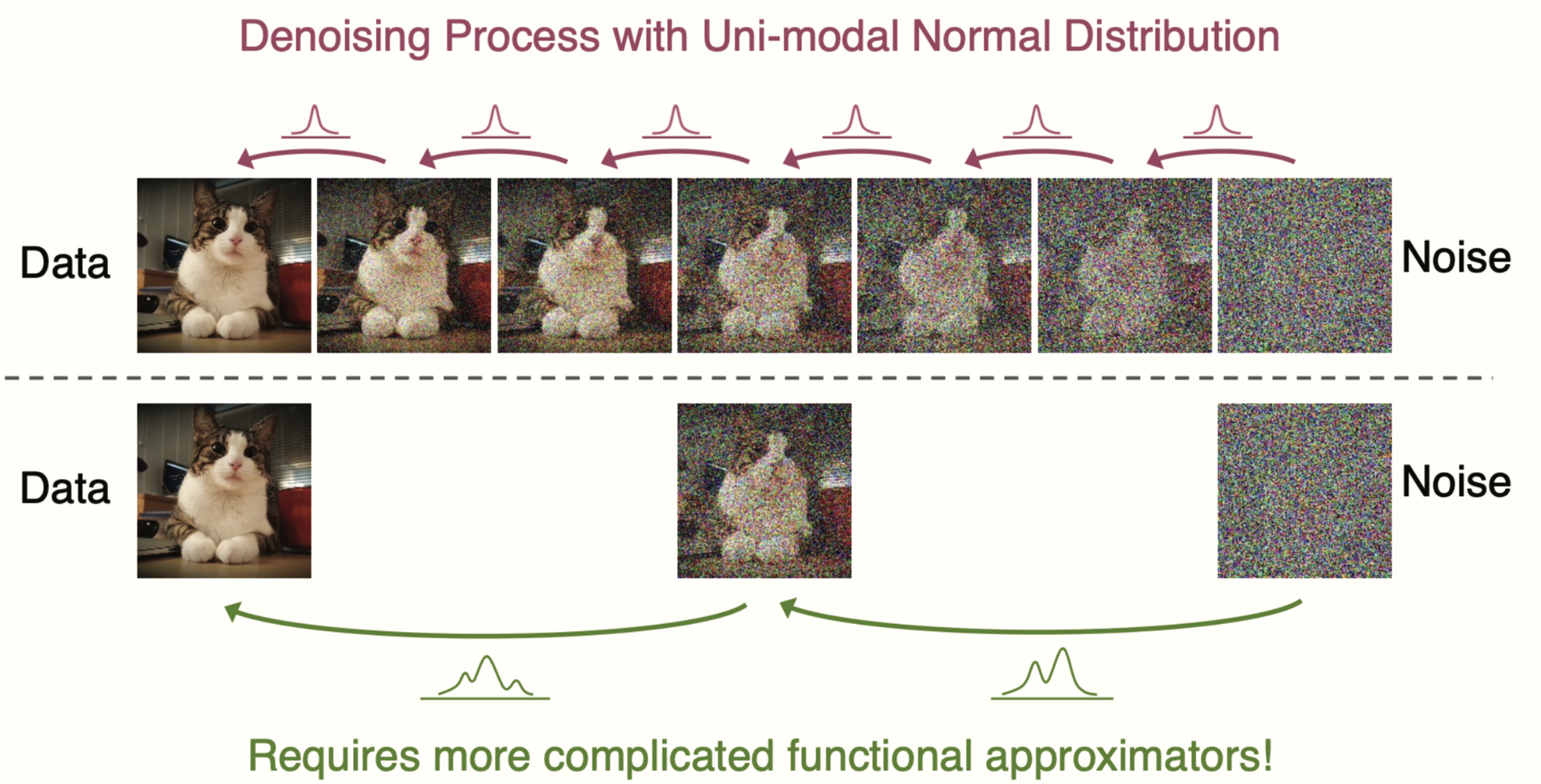
Problem: generating a data point requires T steps.

This is different from GANs and VAEs, which are "single step".

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Skipping steps



Slide source: Kreis et al., via Murphy PMLAT book.

Distilling multiple steps into a new model

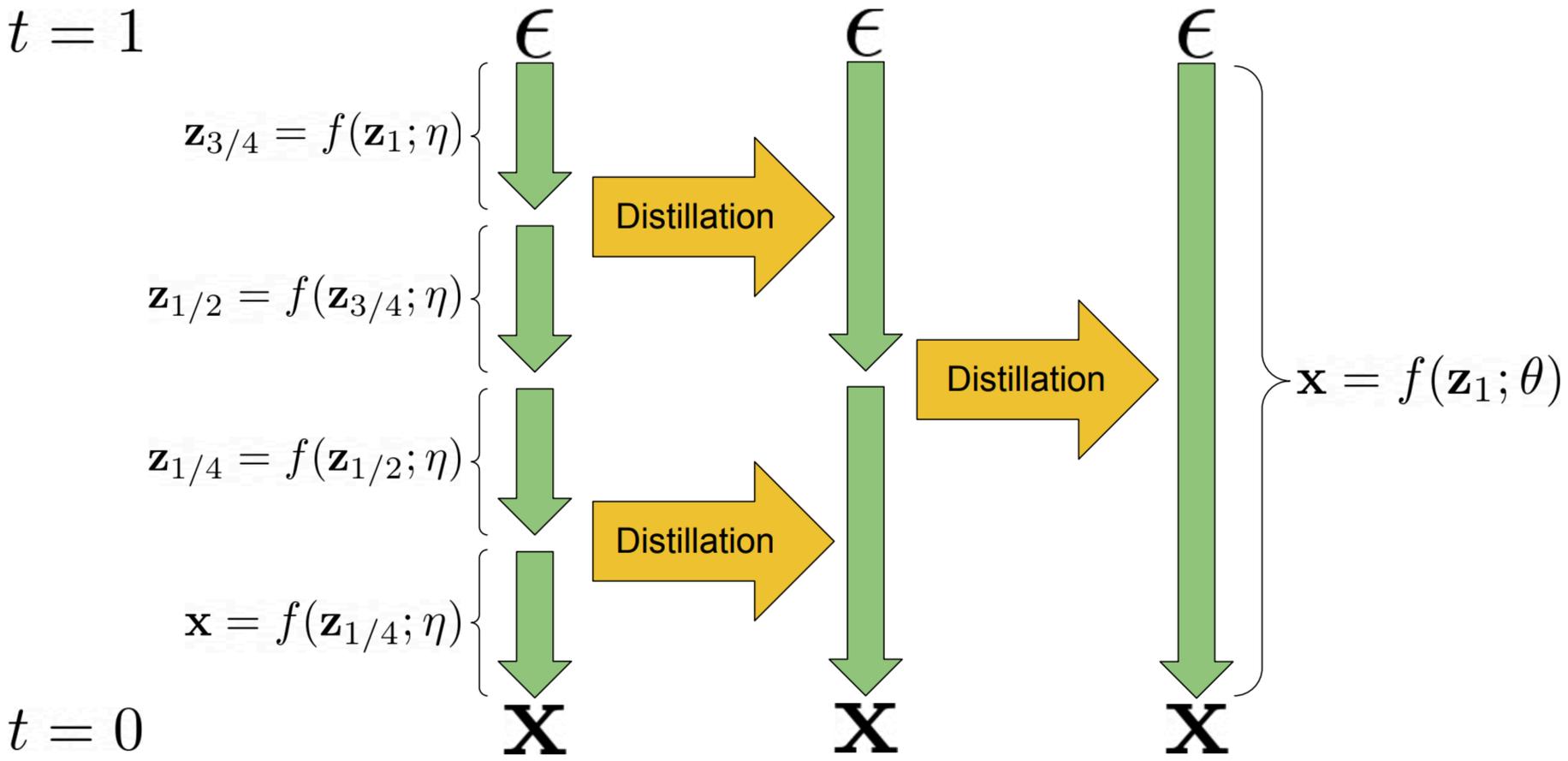


Figure source: [Salimans and Ho, 2022]

Distilling intermediate steps using a GAN

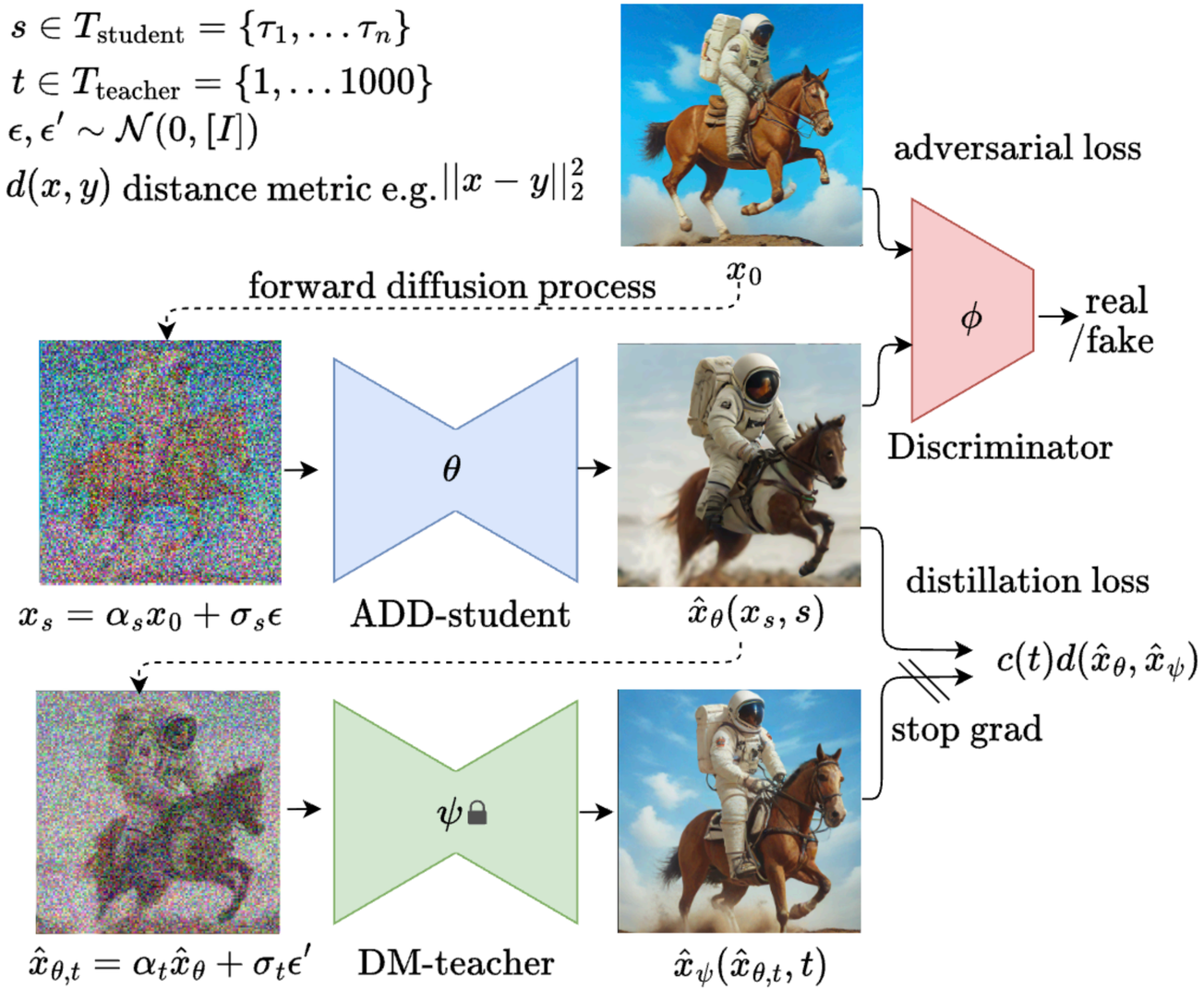


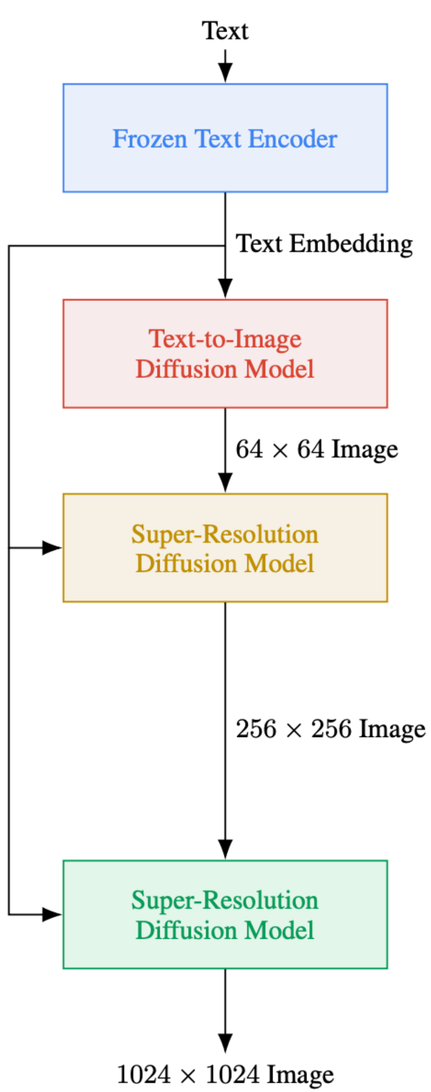
Figure source: [Sauer et al., "Adversarial Diffusion Distillation", 2023]

Distilling intermediate steps using a GAN

Figure source: [Sauer et al., "Adversarial Diffusion Distillation", 2023]

Dimensionality reduction

Multiscale (cascaded) generation.



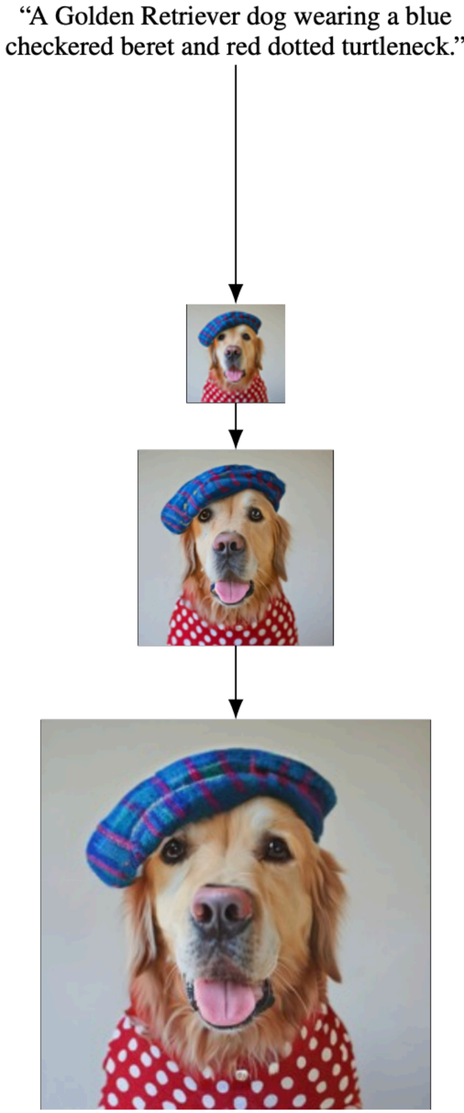
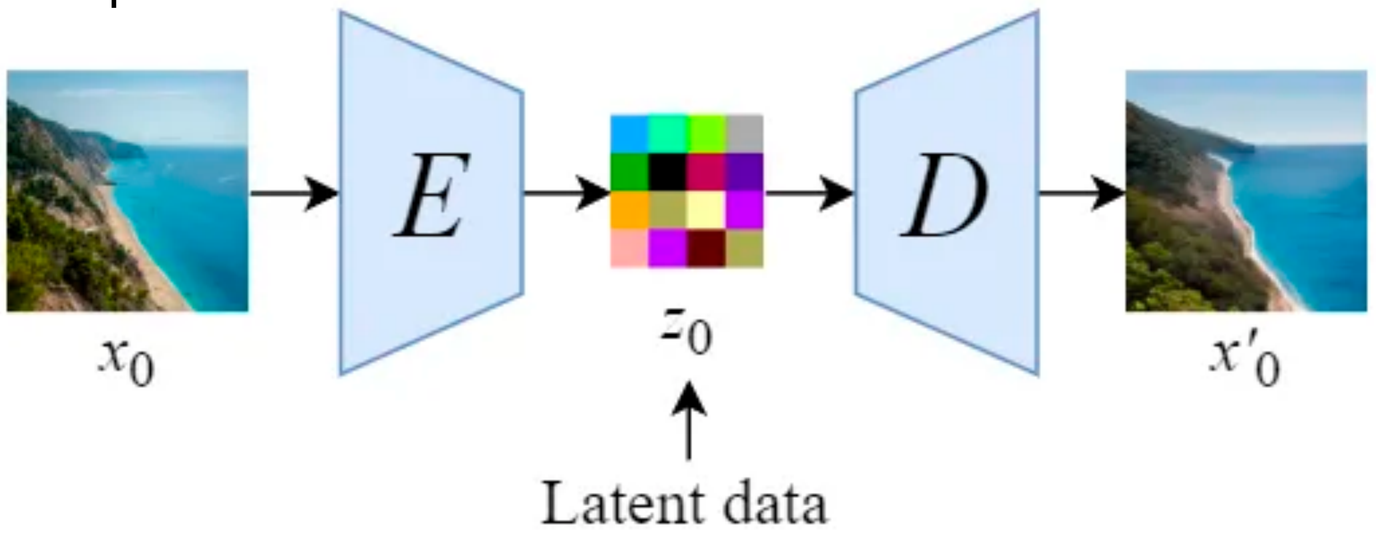


Figure source: [Saharia et al., "Imagen", 2022], via Murphy PMLAT

Latent diffusion models

- Use a separate *encoder* and *decoder* to convert images to and from a lower-dimensional latent space, run diffusion in latent space.
- Lowers input dimensionality, so everything runs faster.
- Another benefit: can design the space to focus on more on "perceptually important" details.

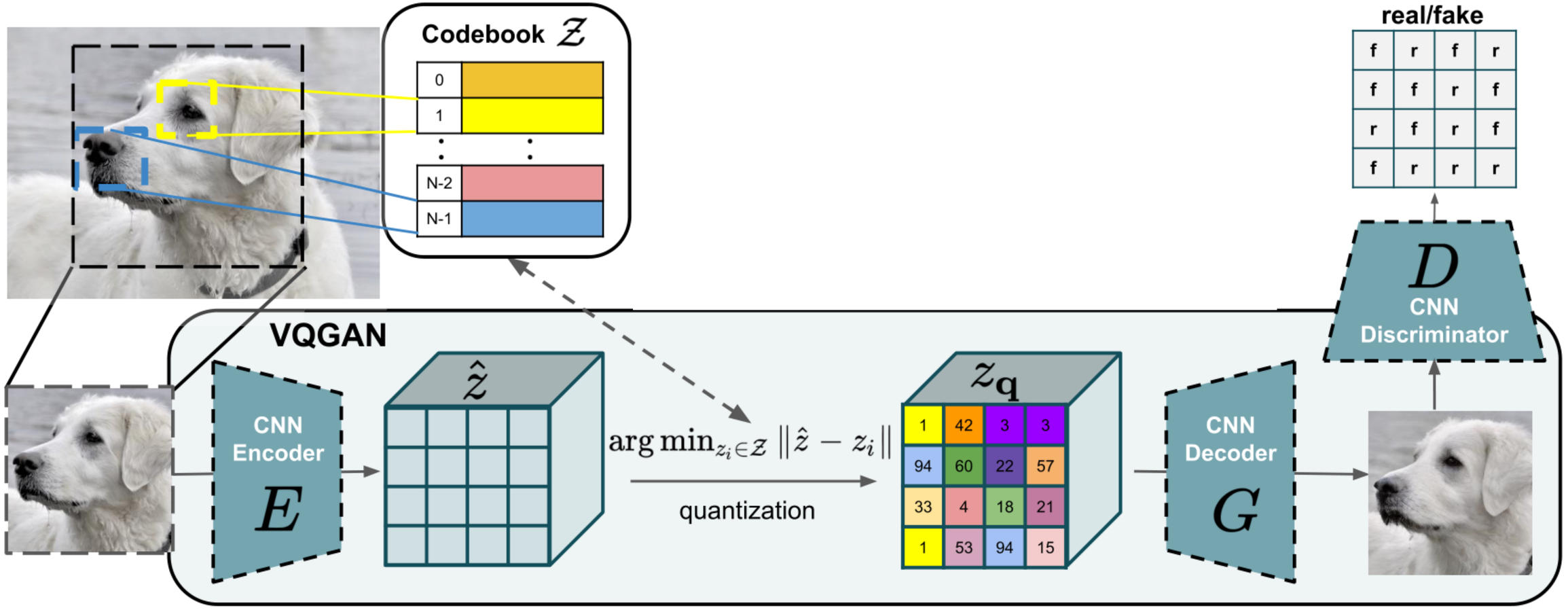


R. Rombach et al. High-Resolution Image Synthesis with Latent Diffusion Models. CVPR 2022

Figure source

Slide source: S. Lazebnik.

Latent space: Patch VAE + adversarial loss



(quantization not typically performed since diffusion is continuous)

[Esser et al., "VQ-GAN", 2021]

Next class: flow matching